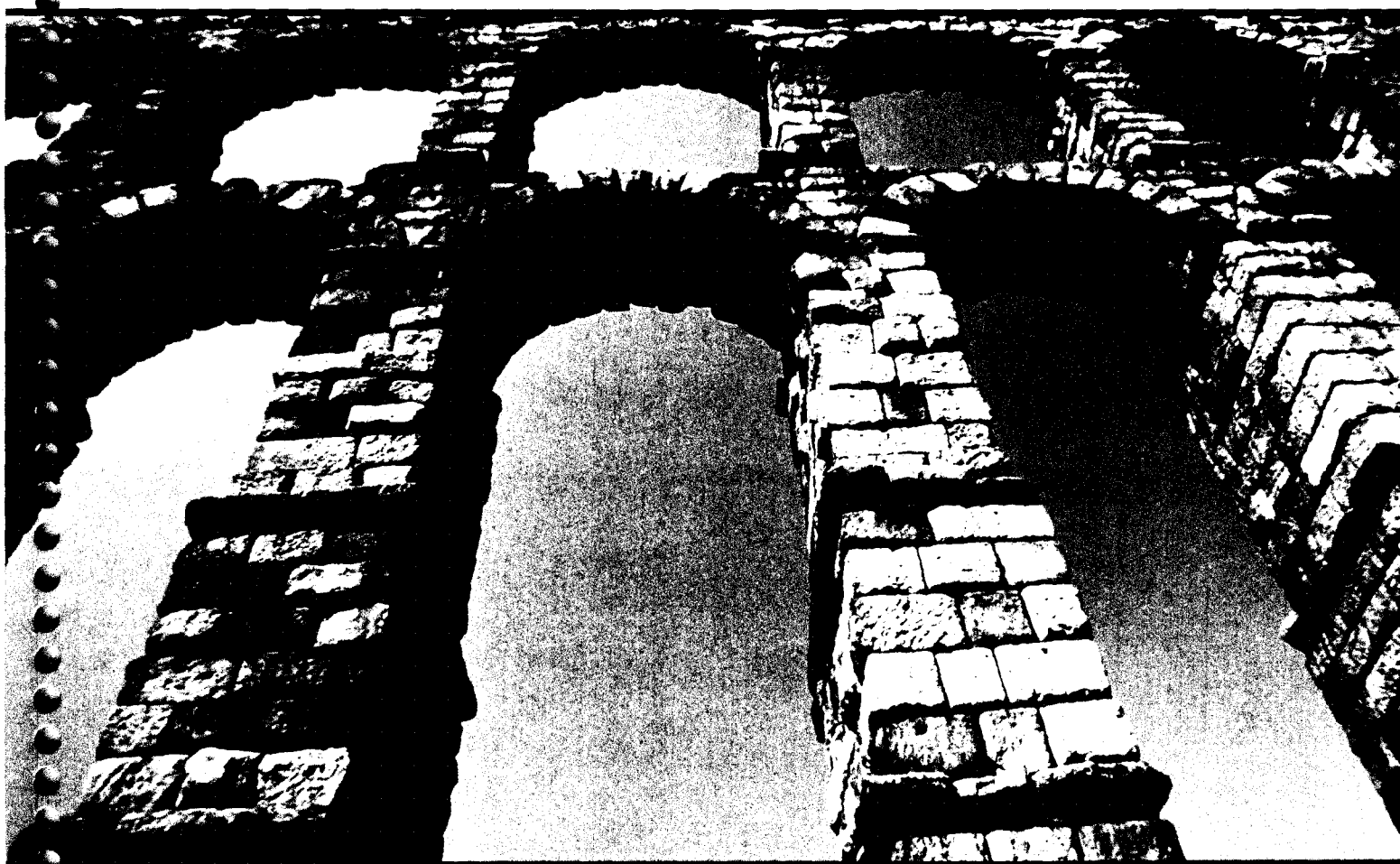
PHP e MySQL
Desenvolvimento Web

Luke Welling
Laura Thomson

Tradução da Terceira Edição

PHP e MySQL
Desenvolvimento Web

Tradução da Terceira Edição

Luke Welling e Laura Thomson

PHP e MySQL
Desenvolvimento Web

Tradução da Terceira Edição

Consultoria Editorial
Lorenzo Ridolfi
Gerente Sênior
Accenture

Sérgio Colcher
Professor do Departamento de Informática
PUC-Rio

Tradução
Docware Traduções Técnicas
e Adriana Kramer

6ª Tiragem

Do original;
PHP and MySQL • Web Development
Tradução autorizada do idioma inglês da edição publicada por Sams Publishing

Editoração Eletrônica
Estúdio Castellani

Copidesque
Michele Gerhardt

Revisão Gráfica
Roberto Mauro Facce

Projeto Gráfico
Elsevier Editora Ltda.
A Qualidade da Informação.
Rua Sete de Setembro, 111 – 16º andar
20050-006 Rio de Janeiro RJ Brasil
Telefone: (21) 3970-9300 FAX: (21) 2507-1991
E-mail: *info@elsevier.com.br*
Escritório São Paulo:
Rua Quintana, 753/8º andar
04569-011 Brooklin São Paulo SP
Tel.: (11) 5105-8555

ISBN 13: 978-85-352-1714-8
ISBN 10: 85-352-1714-2
Edição original: ISBN 0-672-32672-8

Nota: Muito zelo e técnica foram empregados na edição desta obra. No entanto, podem ocorrer erros de digitação, impressão ou dúvida conceitual. Em qualquer das hipóteses, solicitamos a comunicação à nossa Central de Atendimento, para que possamos esclarecer ou encaminhar a questão.

Nem a editora nem o autor assumem qualquer responsabilidade por eventuais danos ou perdas a pessoas ou bens, originados do uso desta publicação.

Central de atendimento
Tel: 0800-265340
Rua Sete de Setembro, 111, 16º andar – Centro – Rio de Janeiro
e-mail: *info@elsevier.com.br*
site: *www.campus.com.br*

CIP-Brasil. Catalogação-na-fonte.
Sindicato Nacional dos Editores de Livros, RJ

W478p

Welling, Luke
PHP e MySQL desenvolvimento Web / Luke Welling, Laura Thomson ; tradução Edson Furmankiewicz e Adriana Kramer. – Rio de Janeiro : Elsevier, 2005 –
6ª Reimpressão.
il.

Tradução de: PHP & MySQL Web development, 3rd ed
Inclui bibliografia
ISBN 85-352-1714-2

1. PHP (Linguagem de programação de computador). 2. SQL (Linguagem de programação de computador). 3. Sites da Web – Desenvolvimento. II. Thomson, Laura. II. Título.

05-0811.   CDD – 005.1
CDU – 004.43

❖

Aos nossos pais

❖

Os Autores

Laura Thomson é conferencista na School of Computer Science and Information Technology da RMIT University em Melbourne, Austrália. Ela é sócia da Tangled Web Design, uma premiada empresa de desenvolvimento para a Web. Laura trabalhou antes para a Telstra e o Boston Consulting Group. Como bacharel em Ciência Aplicada (Ciência da Computação) e bacharel em Engenharia (Engenharia de sistemas de computação) com louvor, atualmente ela está concluindo seu Ph.D. sobre Web sites adaptativos. Nas horas vagas, ela gosta de dormir. É possível entrar em contato com Laura via e-mail: laura@tangledweb.com.au.

Luke Welling é conferencista na School of Computer Science and Information Technology da RMIT University em Melbourne, Austrália. Luke também é sócio da Tangled Web Design. Ele é bacharel em Ciência Aplicada (Ciência da Computação) e está atualmente concluindo o mestrado em algoritmos genéticos na Communication Network Design. Nas horas vagas, ele tenta aperfeiçoar sua insônia. É possível entrar em contato com Luke via e-mail: luke@tangledweb.com.au.

Os Colaboradores

Israel Denis Jr. é um consultor freelance que trabalha em projetos de comércio eletrônico no mundo inteiro. Israel é especializado na integração de pacotes ERP, como SAP e Lawson, com soluções personalizadas para a Web. Quando ele não está ocupado projetando softwares ou escrevendo livros, se diverte viajando à Itália, um lugar que considera seu lar. Israel obteve o mestrado em Engenharia Elétrica pela Georgia Tech em Atlanta, Geórgia, em 1998. Ele é o autor de vários artigos sobre Linux, Apache, PHP e MySQL. Trabalhou para empresas como GE e Procter & Gamble com sistemas de computadores principalmente baseados em UNIX. É possível entrar em contato com Israel via e-mail: idenis@ureach.com.

Chris Newman é programador consultor especializado no desenvolvimento de aplicações dinâmicas de Internet. Ele tem extensa experiência comercial na utilização de PHP e MySQL para produzir um amplo espectro de aplicações para uma base de clientes internacional. Graduado pela Keele University, Chris vive em Stoke-on-Trent, Inglaterra, onde dirige a Lightwood Consultancy Ltd., empresa que fundou em 1999 para ampliar seu interesse pelo desenvolvimento de aplicações para a Internet. Chris ficou fascinado pelo potencial da Internet enquanto estava na universidade e está entusiasmado por trabalhar com tecnologia de ponta. Informações adicionais sobre a Lightwood Consultancy Ltd. podem ser encontradas em http://www.lightwood.net e é possível entrar em contato com Chris pelo e-mail: chris@lightwood.net.

Agradecimentos

Agradecemos à equipe da Sams por todo seu árduo trabalho. Em particular, somos gratos a Shelley Johnston; este livro não teria sido possível sem sua dedicação e paciência. Obrigado também a Israel Denis Jr. e Chris Newman por suas valiosas contribuições.

Agradecemos imensamente o trabalho feito pelas equipes de desenvolvimento de PHP e MySQL. Seu trabalho facilitou por vários anos nossa vida e continua a fazer isso diariamente.

Agradecemos a Adrian Close da eSec por dizer “Vocês podem construir isso no PHP” em 1998. Ele disse que gostaríamos do PHP e parece que estava certo.

Por fim, somos imensamente gratos à nossa família e amigos por nos apoiarem durante o tempo em que fomos anti-sociais. Especificamente, agradecemos o apoio de nossos familiares: Julie, Robert, Martin, Lesley, Adam, Paul, James, Archer e Barton.

Sumário

V Construindo projetos práticos de PHP e MySQL

Introdução

BEM-VINDO AO *PHP e MySQL Desenvolvimento Web.* Nas páginas deste livro, você encontrará um conhecimento destilado de nossas experiências com PHP e MySQL, duas das ferramentas de desenvolvimento Web mais modernas e poderosas que existem.

Nesta introdução, abordaremos:

- Por que você deve ler este livro;
- O que você conseguirá alcançar utilizando este livro;
- O que são o PHP e o MySQL e por que eles são tão bons;
- Uma visão geral dos últimos recursos do PHP 5.0 e MySQL 5.0;
- Como este livro está organizado.

Vamos começar.

Por que você deve ler este livro

Este livro ensina a criar Web sites interativos desde o formulário de pedido mais simples até complexos e seguros sites de comércio eletrônico. Além disso, você aprenderá a fazer isso utilizando tecnologias de código-fonte aberto.

Este livro é dirigido a leitores que já sabem pelo menos os princípios básicos de HTML e têm alguma experiência anterior em uma linguagem de programação moderna, mas não necessariamente programaram para a Internet ou utilizaram um banco de dados relacional. Para programadores iniciantes, este livro também pode ser útil, mas esses precisarão de mais tempo para digeri-lo. Tentamos não omitir nenhum conceito básico, mas a abrangência dos conceitos é realmente rápida. O típico leitor deste livro é alguém que deseja dominar o PHP e o MySQL com o propósito de construir um Web site grande ou comercial. Talvez você já esteja trabalhando em outra linguagem de desenvolvimento Web; se estiver, este livro deve capacitá-lo a trabalhar rapidamente.

Escrevemos a primeira edição deste livro porque estávamos cansados de encontrar livros em PHP que eram basicamente uma referência a funções. Esses livros são úteis, porém não ajudam quando seu patrão ou cliente pede "Construa um carrinho de compras". Fizemos o melhor para tornar cada exemplo útil. Muitos dos exemplos de código podem ser diretamente utilizados no seu Web site e muitos outros podem ser utilizados com pequenas modificações.

O que você conseguirá utilizando este livro

A leitura deste livro capacitará você a construir Web sites dinâmicos do mundo real. Se você construiu Web sites utilizando HTML simples, perceberá as limitações dessa abordagem. O conteúdo estático de um Web site de HTML puro é simplesmente isso – estático. Ele permanece da mesma maneira a menos que você o atualize fisicamente. Os usuários não podem interagir com o site de modo significativo.

Utilizar uma linguagem como PHP e um banco de dados como o MySQL permite tornar os sites dinâmicos, possibilita que eles sejam personalizáveis e contenham informações de tempo real.

Focalizamos deliberadamente aplicações do mundo real, mesmo nos capítulos introdutórios. Começaremos observando um sistema de pedidos on-line simples e apresentaremos pouco a pouco as várias partes do PHP e do MySQL.

Depois, discutiremos os aspectos de comércio eletrônico e segurança relacionados à construção de um Web site real e mostraremos como implementar esses aspectos em PHP e MySQL.

Na seção final deste livro, discutiremos como abordar projetos reais e conduziremos você passo a passo pelo projeto, planejamento e construção dos seguintes projetos:

- Autenticação de usuário e personalização;
- Carrinho de comprass;
- Sistemas de gerenciamento de conteúdo;
- E-mail baseado na Web;
- Gerenciadores de listas de mala-direta;
- Fóruns Web;
- Geração de documento em PDF;
- Conexão a serviços Web com XML e SOAP.

Qualquer um desses projetos deve ser utilizável tal como está ou pode ser modificado para atender às suas necessidades. Escolhemos esses projetos porque acreditamos que eles representam algumas das aplicações baseadas na Web mais comuns construídas por programadores. Mesmo que suas necessidades sejam diferentes, este livro ainda deve ajudá-lo a alcançar seus objetivos sob vários aspectos.

O que é PHP?

O PHP é uma linguagem de criação de scripts do lado do servidor que foi projetada especificamente para a Web. Dentro de uma página HTML, você pode embutir código de PHP que será executado toda vez que a página for visitada. O código de PHP é interpretado no servidor Web e gera HTML ou outra saída que o visitante verá.

O PHP foi concebido em 1994 como resultado do trabalho de uma única pessoa, Rasmus Lerdorf. O PHP foi adotado por outras pessoas inteligentes e foi reescrito três vezes para proporcionar o amplo e aperfeiçoado produto que vemos hoje. Em outubro de 2002, ele era utilizado em mais de nove milhões de domínios em todo o mundo e esse número está crescendo rapidamente. Você pode constatar o número atual em http://www.php.net/usage.php.

O PHP é um produto de código-fonte aberto, o que significa que você tem acesso ao seu código-fonte. É possível utilizá-lo, alterá-lo e redistribuí-lo sem pagar nada.

O PHP significava originalmente *Personal Home Page,* mas foi alterado de acordo com a convenção para atribuição de nomes recursiva do GNU (GNU = GNU's Not Unix) e agora significa *PHP Hypertext Preprocessor.*

A principal versão atual do PHP é a versão 5. Nesta versão, o Zend Engine foi completamente reescrito, e foram feitos aprimoramentos à linguagem.

A home page do PHP está disponível em http://www.php.net.

A home page da Zend Technologies é http://www.zend.com.

O que é MySQL?

O MySQL é um sistema de gerenciamento de banco de dados relacional *(relational database management system – RDBMS)* poderoso e muito rápido. Um banco de dados permite armazenar, pes-

quisar, classificar e recuperar dados de forma eficiente. O servidor de MySQL controla o acesso aos dados para assegurar que vários usuários possam trabalhar com os dados ao mesmo tempo, fornecer acesso rápido aos dados e assegurar que somente usuários autorizados obtenham acesso. Portanto, o MySQL é um servidor multiusuário e multiencadeado (ou *multithreaded*). Ele utiliza *SQL (Structured Query Language)*, a linguagem de consulta padrão de banco de dados em todo o mundo. O MySQL está publicamente disponível desde 1996, mas tem uma história de desenvolvimento que remonta a 1979. O MySQL ganhou o prêmio Journal Readers' Choice Award Linux em várias ocasiões.

MySQL está disponível sob um esquema de licença dupla. Você pode usá-lo sob a licença Open Source (GPL – General Public License) gratuitamente, contanto que cumpra os termos da licença. No entanto, se quiser distribuir uma aplicação não-GPL que inclua o MySQL, você pode comprar uma licença comercial.

Por que utilizar o PHP e o MySQL?

Ao decidir construir um site de comércio eletrônico, há muitos produtos diferentes que podem ser utilizados.

Você precisará escolher:

- O hardware para o servidor Web;
- Um sistema operacional;
- O software do servidor Web;
- Um sistema de gerenciamento de bancos de dados;
- Uma linguagem de programação ou de criação de scripts.

Algumas dessas escolhas dependerão de outras. Por exemplo, nem todos os sistemas operacionais executarão em todo tipo de hardware, nem todas as linguagens de criação de scripts se conectam a todos os bancos de dados e assim por diante.

Neste livro, não prestamos muita atenção ao hardware, ao sistema operacional ou ao software de servidor Web. Não precisamos disso. Uma das maiores qualidades do PHP e do MySQL é o fato de eles funcionarem com qualquer dos maiores sistemas operacionais e muitos dos menores.

Para demonstrar isso, os exemplos neste livro foram escritos e testados nas duas configurações populares:

- Com o Linux utilizando o servidor Web Apache.
- Com o Microsoft Windows XP utilizando o Microsoft Internet Information Server (IIS).

Seja quais forem o hardware, o sistema operacional e o servidor Web escolhidos, acreditamos seriamente que você deve considerar a utilização do PHP e do MySQL.

Algumas capacidades do PHP

Alguns dos principais concorrentes do PHP são Perl, Microsoft ASP.NET, JavaServer Pages (JSP) e ColdFusion.

Em comparação a esses produtos, o PHP tem muitas vantagens, incluindo:

- Alto desempenho;
- Interfaces para muitos sistemas diferentes de banco de dados;
- Bibliotecas integradas para muitas tarefas comuns da Web;
- Baixo custo;

- Facilidade de aprender e utilizar;
- Ótimo suporte orientado a objetos;
- Portabilidade;
- Disponibilidade de código-fonte;
- Disponibilidade de suporte.

A seguir apresentamos uma discussão mais detalhada dessas capacidades.

Desempenho

O PHP é muito eficiente. Utilizando um único servidor barato, você pode atender a milhões de acessos por dia. Os benchmarks publicados pela Zend Technologies (http://www.zend.com) mostram o PHP superando o desempenho do concorrente.

Integração de banco de dados

O PHP tem conexões nativas disponíveis para muitos sistemas de banco de dados. Além do MySQL, você pode conectar-se diretamente a bancos de dados PostgreSQL, mSQL, Oracle, dbm, FilePro, HyperWave, Informix, InterBase e Sybase, entre outros. O PHP 5 também tem uma interface integrada SQL para um arquivo simples, chamada SQLite.

Utilizando o *Open Database Connectivity Standard (ODBC)*, você pode conectar-se a qualquer banco de dados que forneça um driver de ODBC. Isso inclui produtos da Microsoft e muitos outros.

Bibliotecas integradas

Como o PHP foi projetado para utilização na Web, ele tem muitas funções integradas para realizar muitas tarefas úteis relacionadas à Web. Você pode gerar imagens GIF instantâneas, conectar-se a outros serviços de rede, enviar e-mail, trabalhar com cookies e gerar documentos PDF, tudo com apenas algumas linhas de código.

Custo

O PHP é gratuito. Você pode fazer download da última versão a qualquer momento em *http://www.php.net*, gratuitamente.

Aprendizagem do PHP

A sintaxe do PHP está baseada em outras linguagens de programação, principalmente C e Perl. Se já conhece Perl ou C ou uma linguagem do tipo C, como C++ ou Java, você conseguirá produzir quase imediatamente utilizando o PHP.

Suporte orientado a objetos

A versão 5 do PHP possui recursos orientados a objetos muito bem elaborados. Se você aprendeu a programar em Java ou C++, encontrará os recursos (e geralmente a sintaxe) esperados, tal com herança, atributos e métodos privados e protegidos, classes e métodos abstratos, interfaces, construtores e destruidores. Você até encontrará recursos menos comuns, como o comportamento de iteração integrado. Algumas dessas funcionalidades já estavam disponíveis no PHP versões 3 e 4, mas o suporte orientado a objetos na versão 5 é muito mais completo.

Portabilidade

O PHP está disponível para muitos sistemas operacionais diferentes. Você pode escrever código de PHP em sistemas operacionais do tipo Unix gratuitos como o Linux e o FreeBSD, versões comerciais do Unix como Solaris e IRIX, ou em versões diferentes do Microsoft Windows.

Código bem escrito normalmente funcionará sem modificação em um sistema diferente executando o PHP.

Código-fonte

Você tem acesso ao código-fonte do PHP. Ao contrário dos produtos comerciais e de código-fonte fechado, se houver algo que deseje modificar ou adicionar à linguagem, você é livre para fazer isso.

Não é preciso esperar o fabricante lançar os patches e nem se preocupar com a possibilidade de o fabricante largar o negócio ou decidir parar de oferecer suporte ao produto.

Disponibilidade de suporte

A Zend Technologies (www.zend.com), a empresa reponsável pelo mecanismo que mantém o PHP, financia o desenvolvimento do PHP oferecendo suporte e softwares relacionados numa base comercial.

O que há de novo no PHP 5.0?

É provável que recentemente você tenha mudado para o PHP 5.0 a partir das versões do PHP 4.x. Como é de se esperar de uma nova versão, ela possui algumas mudanças significativas. O mecanismo da Zend por trás do PHP foi reescrito para esta versão. Os principais recursos novos são:

- Melhor suporte orientado a objetos construído a partir de um modelo de objeto completamente novo (ver Capítulo 6).
- Exceções para manipulação de erros escaláveis e sustentáveis (ver Capítulo 7).
- SimpleXML para fácil manipulação de dados XML (ver Capítulo 33).

Outras mudanças incluem a remoção de algumas extensões da instalação padrão do PHP para a biblioteca PECL, o aprimoramento do suporte a streams e a adição do SQLite.

Algumas capacidades do MySQL

Alguns dos principais concorrentes do MySQL são PostgreSQL, Microsoft SQL Server e Oracle.

MySQL tem muitas capacidades, incluindo:

- Alto desempenho;
- Baixo custo;
- Fácil configuração e aprendizado;
- Portabilidade;
- Disponibilidade do código-fonte;
- Disponibilidade de suporte.

A seguir apresentamos uma discussão mais detalhada dessas capacidades.

Desempenho

O MySQL é inegavelmente rápido. Você pode ver a página de benchmark dos desenvolvedores em *http://web.mysql.com/benchmark.html*. Muitos desses benchmarks mostram como o MySQL é bem mais rápido do que os concorrentes. Em 2002, a *eWeek* publicou um benchmark comparando cinco bancos de dados que fazem uma aplicação Web funcionar. O melhor resultado foi um empate entre o MySQL e o Oracle, que é muito mais caro.

Baixo custo

O MySQL está disponível sem nenhum custo, sob uma licença de código-fonte aberto, ou a baixo custo, sob uma licença comercial. A licença é necessária se você quiser redistribuir o MySQL como parte de uma aplicação e não deseja disponibilizar sua aplicação com uma licença de código-fonte aberto. Se não pretende distribuir sua aplicação ou se está trabalhando em um software livre, não precisa comprar a licença.

Facilidade de uso

Os bancos de dados mais modernos utilizam o SQL. Se você utilizou outro RDBMS, não deve ter nenhum problema de adaptação com esse. O MySQL também é mais fácil de configurar que muitos produtos semelhantes.

Portabilidade

O MySQL pode ser utilizado em muitos sistemas Unix diferentes, bem como sob o Microsoft Windows.

Código-fonte

Assim como ocorre com o PHP, você pode obter e modificar o código-fonte para o MySQL. Na maior parte do tempo, esse aspecto não é importante para a maioria dos usuários, mas oferece a você bastante tranqüilidade, garantindo uma continuidade futura e fornecendo opções para uma emergência.

Disponibilidade de suporte

Nem todos os produtos de código-fonte aberto possuem uma empresa mãe que ofereça suporte, treinamento, consultoria e certificação, mas todos esses benefícios são oferecidos pela MySQL AB (*www.mysql.com*).

O que há de novo no MySQL 5.0?

As principais mudanças incluídas no MySQL 5.0 incluem:

- Procedimentos armazenados (ver Capítulo 13);
- Suporte a cursores.

Outras mudanças incluem maior concordância com os padrões ANSI e aprimoramentos de velocidade. Se você ainda está utilizando uma versão antiga 4.x ou 3.x do servidor MySQL, precisa saber que novos recursos foram adicionados às diversas versões, desde a 4.0:

- Suporte a subconsulta;
- Tipos GIS para armazenar dados geográficos;

- Suporte aprimorado para internacionalização;
- O mecanismo de armazenamento de transação segura InnoDB incluído como padrão;
- O cache de consulta MySQL, que melhora bastante a velocidade de consultas repetitivas executadas com freqüência por aplicações Web

Como este livro está organizado?

Este livro é dividido em cinco partes principais.

A Parte I fornece uma visão geral das principais partes da linguagem PHP com exemplos. Cada um deles será um exemplo real utilizado na construção de um site de comércio eletrônico, em vez de código de "brinquedo". Abordaremos esse assunto no Capítulo 1. Se já utilizou o PHP, você pode dar apenas uma passada de olhos nesse capítulo. Se for um iniciante em PHP ou programação, talvez queira passar um pouco mais tempo nela. Mesmo que tenha familiaridade com o PHP, você vai querer ler o Capítulo 6 pois a funcionalidade orientada a objeto mudou de forma significativa no PHP 5.

A Parte II discute o projeto e os conceitos envolvidos na utilização dos sistemas de banco de dados relacional como MySQL, utilização do SQL, conexão do banco de dados MySQL ao mundo com o PHP, e discute tópicos avançados de MySQL, como segurança e otimização.

A Parte III abrange algumas questões gerais envolvidas no desenvolvimento de um site de comércio eletrônico que utilize qualquer linguagem. A mais importante dessas questões é a segurança. Então, discutimos como você pode utilizar o PHP e o MySQL para autenticar usuários e reunir, transmitir e armazenar dados de modo seguro.

A Parte IV oferece uma explicação detalhada de algumas das funções integradas importantes em PHP. Selecionamos grupos de funções que são provavelmente úteis ao construir um site de comércio eletrônico. Você aprenderá a interação com o servidor e com a rede, a geração de imagens, a manipulação de data e hora e as variáveis de sessão.

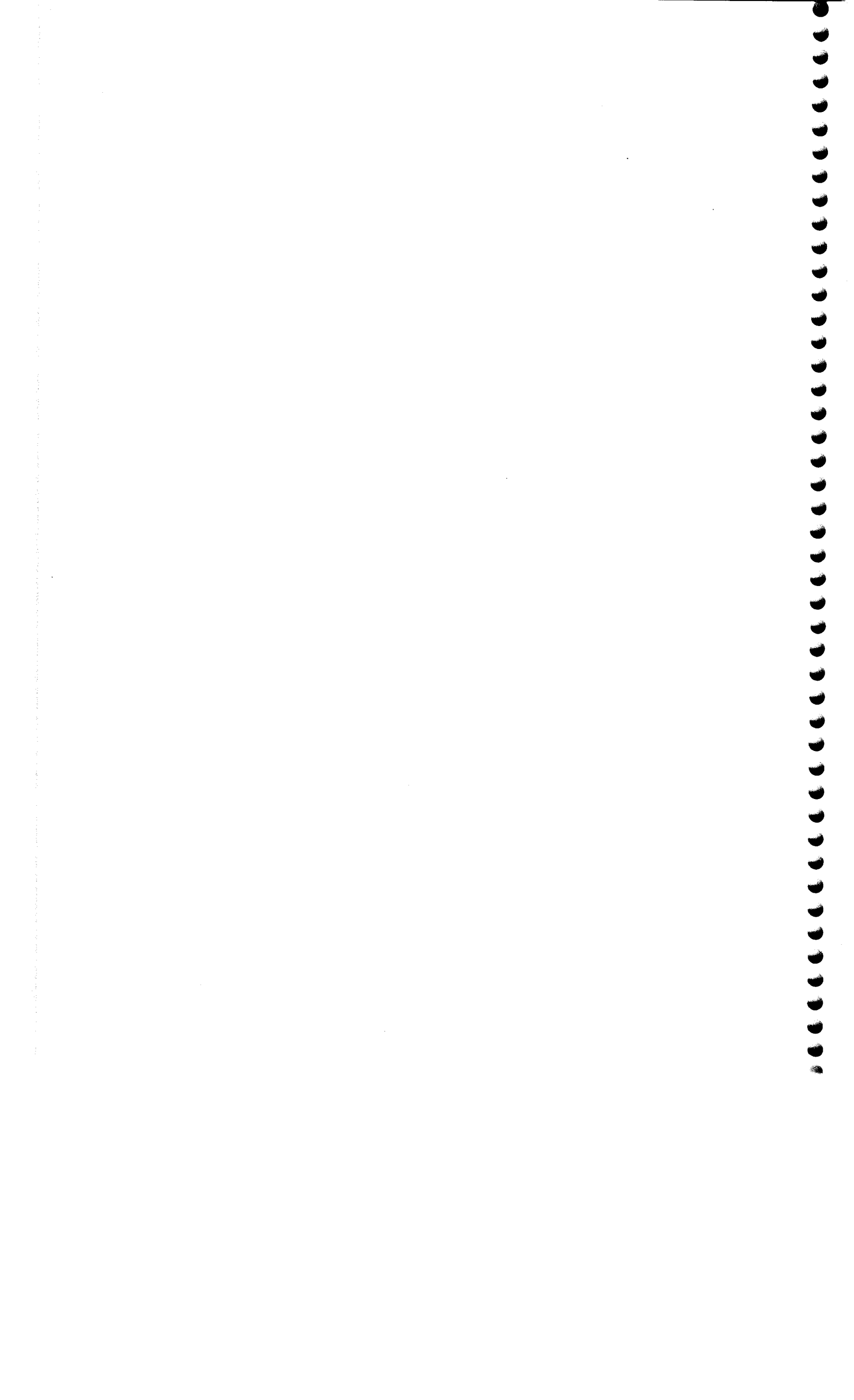
A Parte V lida com questões práticas do mundo real como gerenciamento de grandes projetos e depuração, e fornece exemplos de projetos que demonstram a potência e a versatilidade do PHP e do MySQL.

Para finalizar

Esperamos que você goste deste livro e de aprender o PHP e o MySQL tanto quanto nós quando utilizamos pela primeira vez esses produtos. Eles realmente são muito agradáveis. Logo, você poderá se unir aos milhares de desenvolvedores Web que utilizam essas ferramentas poderosas para facilmente construir Web sites dinâmicos de tempo real.

I

Utilizando o PHP

1

Visão geral do PHP

ESTE CAPÍTULO FORNECE UMA RÁPIDA VISÃO GERAL da sintaxe do PHP e das construções da linguagem. Se você já é um programador de PHP, talvez este capítulo preencha algumas lacunas no seu conhecimento. Se você já tiver uma base sobre a utilização de ASP, C ou outra linguagem de programação, este capítulo o ajudará a familiarizar-se com o assunto rapidamente.

Neste livro, você aprenderá a utilizar o PHP trabalhando passo a passo com uma grande quantidade de exemplos do mundo real, extraídos de nossa experiência na construção de sites de comércio eletrônico. Com freqüência, os textos de programação ensinam sintaxe básica com exemplos muito simples. Optamos por não fazer isso. Reconhecemos que freqüentemente o que você deseja fazer é algo prático e funcional, entender como a linguagem é utilizada, em vez de estudar arduamente mais outra referência de sintaxe e função que nem chega a ser melhor que o manual on-line.

Experimente os exemplos – digite-os ou carregue-os a partir do CD-ROM, altere-os, divida-os em partes e aprenda a corrigi-los novamente.

Neste capítulo, iniciaremos com o exemplo de um formulário de pedido on-line de produtos para aprender como as variáveis, as expressões e os operadores são utilizados no PHP. Também abordaremos tipos de variáveis e a precedência de operadores. Você aprenderá a acessar variáveis de formulário e a manipulá-las calculando o preço total e os impostos em um pedido de cliente.

Então desenvolveremos o exemplo de formulário de pedido on-line utilizando nosso script de PHP para validar os dados de entrada. Examinaremos o conceito de valores booleanos e daremos exemplos do uso de `if`, `else`, do operador `?:` e da instrução `switch`. Por fim, exploramos os loops escrevendo um pequeno código em PHP para gerar tabelas de HTML repetitivas.

Os tópicos-chave que você aprenderá neste capítulo incluem:

- Embutindo PHP em HTML;
- Adicionando conteúdo dinâmico;
- Acessando variáveis de formulário;
- Identificadores;
- Variáveis declaradas pelo usuário;
- Tipos de variável;
- Atribuindo valores a variáveis;
- Constante;
- Escopo de variável;
- Operadores e precedência;
- Expressões;

- Funções de variável;
- Tomando decisão com `if`, `else` e `switch`;
- Iteração: `while`, `do` e loops `for`.

Utilizando o PHP

A fim de lidar com os exemplos neste capítulo e no restante do livro, você precisará acessar um servidor Web com o PHP instalado. Para aproveitar ao máximo os exemplos e estudos de caso, você deve executá-los e tentar alterá-los. Para fazer isso, você precisará de um ambiente de testes em que possa testar seu trabalho.

Se o PHP não estiver instalado na máquina, você precisará começar instalando-o, ou fazer com que o administrador de sistema o instale para você. O Apêndice A contém instruções para instalação. Tudo o que você precisa para instalar o PHP no UNIX ou no Windows pode ser encontrado no CD-ROM que acompanha este livro.

Criando uma aplicação de exemplo: Bob's Auto Parts

Uma das aplicações mais comuns de qualquer linguagem de criação de scripts do lado servidor é processar formulários de HTML. Você começará aprendendo o PHP implementando um formulário de pedido para a empresa Bob's Auto Parts, uma empresa fictícia de autopeças. Todo o código para os exemplos do Bob utilizados neste capítulo está no diretório chamado `chapter01` no CD-ROM.

Criando um formulário de pedido

A esta altura, o programador de HTML do Bob já conseguiu até configurar um formulário de pedido para as peças que Bob vende. Esse formulário de pedido relativamente simples, mostrado na Figura 1.1, é semelhante a muitos que você provavelmente viu ao navegar pela Internet. A primeira informação que Bob gostaria de obter é o que seu cliente encomendou, o total do pedido e o valor do imposto sobre vendas que deve ser acrescido ao pedido.

Figura 1.1 O formulário de pedido inicial do Bob registra somente produtos e quantidades.

Parte da HTML para isso é mostrada na Listagem 1.1. Há dois pontos importantes a serem notados neste código.

Listagem 1.1 **orderform.html** – **A HTML para o formulário básico de pedido do Bob**

```
<form action="processorder.php" method=post>
<table border=0>
<tr bgcolor=#cccccc>
  <td width=150>Item</td>
  <td width=15>Quantity</td>
</tr>
<tr>
  <td>Tires</td>
  <td align="center"><input type="text" name="tireqty" size="3"
    maxlength="3"></td>
</tr>
<tr>
  <td>Oil</td>
  <td align="center"><input type="text" name="oilqty" size="3" maxlength="3"></td>
</tr>
<tr>
  <td>Spark Plugs</td>
  <td align="center"><input type="text" name="sparkqty" size="3"
    maxlength="3"></td>
</tr>
<tr>
  <td colspan="2" align="center"><input type="submit" value="Submit Order"></td>
</tr>
</table>
</form>
```

O primeiro ponto a notar é que configuramos a ação do formulário para ser o nome do script de PHP que processará o pedido do cliente. (Escreveremos esse script em breve.) Em geral, o valor do atributo `action` é o URL que será carregado quando o usuário pressionar o botão Submit. Os dados que o usuário digitou no formulário serão enviados para esse URL via método especificado no atributo `method`, que pode ser `get` (acrescentado ao fim do URL) ou `post` (enviado como um pacote separado).

O segundo ponto que você deve notar são os nomes dos campos de formulário – `tireqty`, `oilqty` e `sparkqty`. Utilizaremos esses nomes novamente no nosso script de PHP. Por isso, é importante dar nomes significativos aos campos de formulário dos quais você possa facilmente se lembrar ao começar a escrever o script de PHP. Alguns editores HTML gerarão nomes de campo como `field23` por padrão. Esses nomes são difíceis de lembrar. Sua vida como programador de PHP será mais fácil se esses nomes refletirem os dados que serão digitados no campo.

Talvez você queira considerar adotar um padrão de codificação de nomes de campo para que todos os nomes de campo no site inteiro utilizem o mesmo formato. Isso facilita lembrar, por exemplo, se você abreviou uma palavra em um nome de campo ou colocou sublinhado no lugar de um espaço.

Processando o formulário

Para processar o formulário, precisaremos criar o script mencionado no atributo ACTION da tag `form` chamada `processorder.php`. Abra seu editor de textos e crie esse arquivo. Digite o seguinte código:

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
```

```
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
</body>
</html>
```

Observe como tudo que digitamos até agora é apenas HTML simples. Agora é hora de adicionar algum código simples de PHP ao nosso script.

Embutindo PHP na HTML

Sob o título <h2> no arquivo, adicione as seguintes linhas:

```
<?php
  echo '<p>Order processed.</p>';
?>
```

Salve o arquivo e o carregue no navegador preenchendo o formulário do Bob e clicando no botão Submit Order. Você deve ver algo semelhante à saída mostrada na Figura 1.2.

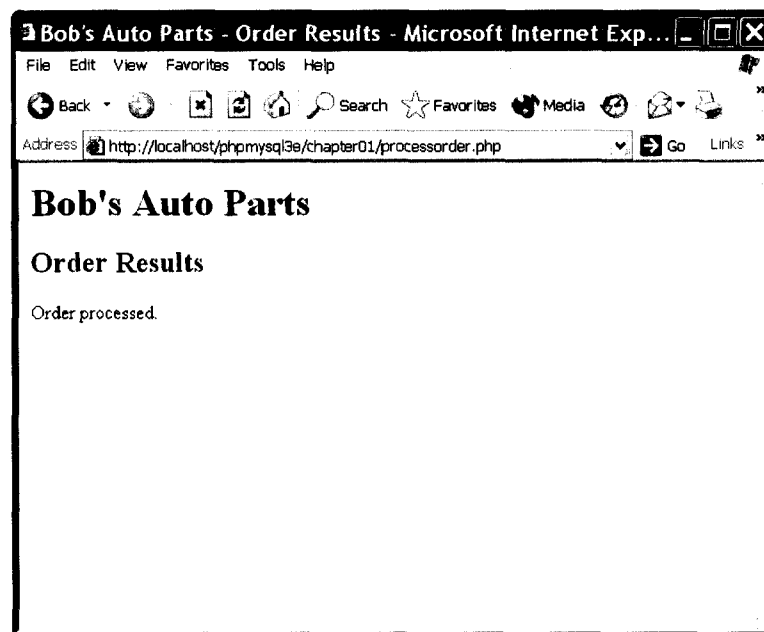



Figura 1.2 O texto passado para construção echo do PHP é ecoado para o navegador.

Note como o código de PHP que escrevemos foi embutido dentro de um arquivo HTML de aparência normal. Tente visualizar o código-fonte do navegador. Você deve ver este código:

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<p>Order processed.</p></body>
</html>
```

Nada do PHP bruto é visível. Isso ocorre porque o interpretador de PHP executou por meio do script e o substituiu pela saída do script. Isso significa que a partir do PHP podemos produzir

HTML limpa e visualizável em qualquer navegador – em outras palavras, o navegador do usuário não precisa entender PHP.

Isso ilustra o conceito da criação de script do lado do servidor. O PHP foi interpretado e executado no servidor Web, ao contrário do JavaScript e outras tecnologias do lado do cliente que são interpretadas e executadas dentro de um navegador Web na máquina do usuário.

O código que agora temos nesse arquivo consiste em quatro partes:

- HTML;
- Tags de PHP;
- Instruções de PHP;
- Espaços em branco.

Também podemos acrescentar:

- Comentários.

A maioria das linhas no exemplo é apenas HTML simples.

Utilizando tags de PHP

O código de PHP no exemplo anterior iniciava com `<?php` e terminava com `?>`. Isso é semelhante às tags HTML porque todas iniciam com um símbolo de menor que (<) e terminam com um símbolo de maior que (>). Esses símbolos são chamados tags de PHP que dizem ao servidor Web onde o código de PHP inicia e onde termina. Qualquer texto entre as tags será interpretado como PHP. Qualquer texto fora dessas tags será tratado como HTML normal. As tags de PHP permitem *escapar* da HTML.

Há diferentes tipos de tag disponíveis. Vejamos isso em mais detalhe.

Estilos de tag do PHP

Há na realidade quatro estilos diferentes de tags de PHP que podem ser utilizados. Cada um dos seguintes fragmentos de código é equivalente.

- Estilo XML

  ```
  <?php echo '<p>Order processed.</p>'; ?>
  ```

  Esse é o estilo de tag que será utilizado neste livro. É o estilo preferido de tag para utilizar com o PHP. O administrador do servidor não pode desativá-lo; portanto, você pode garantir que ele estará disponível em todos os servidores. Esse estilo de tag pode ser utilizado com documentos XML (Extensible Markup Language). Se planeja empregar XML no seu site, definitivamente você deve utilizar esse estilo de tag.

- Estilo abreviado

  ```
  <? echo '<p>Order processed.</p>'; ?>
  ```

  Esse estilo de tag é o mais simples e segue o estilo de uma instrução de processamento SGML (Standard Generalized Markup Language). Utilizando esse tipo de tag – que é o mais fácil de digitar – você precisa ativar `short_open_tag` no arquivo config ou compilar o PHP com tags abreviadas ativadas. Você pode encontrar informações adicionais sobre a instalação no Apêndice A. O uso desse estilo não é recomendado porque, apesar de estar ativado por padrão, os administradores de sistema costumam desativá-lo pois interfere com as declarações de documento XML.

- Estilo SCRIPT

```
<script language='php'> echo '<p>Order processed.</p>'; </script>
```

Esse estilo de tag é o mais longo e será familiar se você já tiver utilizado JavaScript ou VBScript. Ele pode ser empregado no caso de você estar utilizando um editor HTML que apresente problemas com os outros estilos de tag.

- Estilo ASP

```
<% echo '<p>Order processed.</p>'; %>
```

Esse estilo de tag é o mesmo utilizado em Active Server Pages (ASP). Ele pode ser utilizado se você ativou a definição de configuração `asp_tags`. É possível que você deseje utilizar esse estilo de tag se estiver utilizando um editor projetado para ASP ou ASP.NET ou se você já programa em ASP ou ASP.NET. Observe que, por padão, esse estilo de tag está desativado.

Instruções de PHP

Dizemos ao interpretador do PHP o que fazer inserindo as instruções do PHP entre as tags de abertura e fechamento. Neste exemplo, usamos somente um tipo de instrução:

```
echo '<p>Order processed.</p>';
```

Como provavelmente você adivinhou, utilizar a construção `echo` tem um resultado muito simples; ela imprime (ou ecoa) para o navegador a string passada a ela. Na Figura 1.2, vemos que o resultado é o texto `Order processed.` aparecendo na janela de navegador.

Você notará que um ponto-e-vírgula aparece no final da instrução `echo`. Isso é utilizado para separar instruções em PHP de modo muito semelhante a um ponto que é utilizado para separar frases na língua escrita normal. Se você programou em C ou Java antes, saberá utilizar o ponto-e-vírgula dessa maneira.

Esquecer de usar o ponto-e-vírgula é um erro de sintaxe comum que é facilmente cometido. Mas é igualmente fácil localizá-lo e corrigi-lo.

Espaços em branco

Os caracteres de espaçamento como novas linhas (retornos de carro), espaços e tabulações são conhecidos como espaços em branco. Como você provavelmente já sabe, os navegadores ignoram espaços em branco em HTML. O mecanismo do PHP também. Considere estes dois fragmentos de HTML:

```
<h1>Welcome to Bob's Auto Parts!</h1><p>What would you like to order today?</p>
```

e

```
<h1>Welcome           to Bob's
Auto Parts!</h1>
<p>What would you like
to order today?</p>
```

Esses dois trechos de código HTML produzem saída idêntica porque aparecem como sendo idênticos para o navegador. Entretanto, você pode e é encorajado a utilizar espaços em branco na HTML como um auxílio aos humanos – para aprimorar a legibilidade do código HTML. O mesmo é verdadeiro para PHP. Não há nenhuma necessidade de ter qualquer espaço em branco entre instruções de PHP, mas torna o código mais fácil de ler se colocarmos cada instrução em uma linha separada. Por exemplo,

```
echo 'hello ';
echo 'world';
```

e

```
echo 'hello ';echo 'world';
```

são equivalentes, porém a primeira versão é mais fácil de ler.

Comentários

Os comentários são exatamente o que seu nome diz: comentários no código atuam como notas para as pessoas que lêem o código. Os comentários podem ser utilizados para explicar o propósito do script, quem escreveu, por que foi escrito de tal maneira, quando foi modificado pela última vez e assim por diante. Você geralmente localizará comentários em quase todos os scripts exceto nos mais simples do PHP.

O interpretador de PHP ignorará qualquer texto em um comentário. Essencialmente, o analisador de sintaxe do PHP pula os comentários que são equivalentes a espaços em branco.

O PHP suporta comentários no estilo do C, do C++ e do script de shell.

Esse é um comentário de diversas linhas no estilo do C que talvez apareça no início de nosso script de PHP:

```
/* Autor: Bob Smith
   Última modificação: 10 de abril
   Esse script processa os pedidos de cliente.
*/
```

Comentários de diversas linhas devem iniciar com um /* e terminar com */. Como em C, os comentários de diversas linhas não podem ser aninhados.

Você também pode utilizar comentários de uma única linha, tanto no estilo do C++:

```
echo '<p>Order processed.</p>'; // Inicia a impressão do pedido
```

como no estilo de script de shell:

```
echo '<p>Order processed.</p>'; # Inicia a impressão do pedido
```

Com ambos esses estilos, tudo depois do símbolo de comentário (# ou //) é um comentário até alcançarmos o final da linha ou a tag de fechamento do PHP, o que vier primeiro.

Na linha de código seguinte, o texto antes da tag de fechamento, `isto é um comentário`, faz parte de um comentário. O texto depois da tag de fechamento, `isto não`, será tratado como HTML porque está fora da tag de fechamento:

```
// isto é um comentário ?> isto não
```

Adicionando conteúdo dinâmico

Até agora, não utilizamos o PHP para fazer nada que não pudéssemos fazer com HTML simples.

A principal razão de utilizar uma linguagem de criação de scripts do lado do servidor é ser capaz de fornecer conteúdo dinâmico para usuários de um site. Essa é uma aplicação importante porque isso muda o conteúdo de acordo com as necessidades de um usuário ou ao longo do tempo manterá os visitantes voltando para um site. O PHP permite fazer isso facilmente.

Vamos iniciar com um exemplo simples. Substitua o PHP em `processorder.php` pelo seguinte código:

```
<?php
  echo '<p>Order processed at ';
  echo date('H:i, jS F');
```

```
  echo '</p>';
?>
```

Nesse código, estamos utilizando a função date( ) predefinida do PHP para dizer ao cliente a data e a hora em que o pedido foi processado. Isso será diferente toda vez que o script for executado. A saída de execução do script em uma ocasião é mostrada na Figura 1.3.

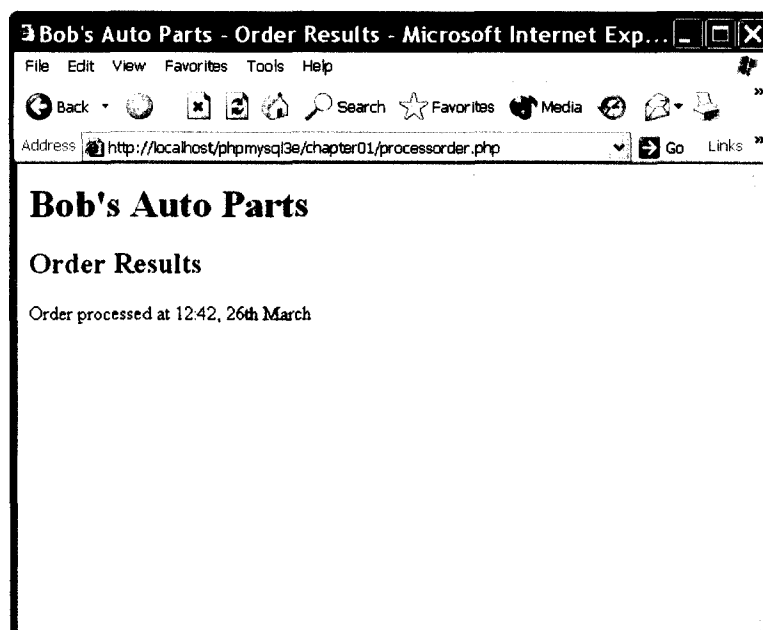



Figura 1.3 A função date( ) do PHP retorna uma string de data formatada.

Chamando funções

Veja a chamada a date( ). Essa é a forma geral que as chamadas de função assumem. O PHP tem uma extensa biblioteca de funções que você pode utilizar ao desenvolver aplicações Web. A maioria dessas funções precisa receber e retornar alguns dados.

Veja a chamada de função:

```
date('H:i, jS F')
```

Observe que estamos passando uma string (dados de texto) para a função dentro de um par de parênteses. Isso é chamado de *argumento* ou *parâmetro* da função. Esses argumentos são a entrada utilizada pela função para dar saída a alguns resultados específicos.

Usando a função date( )

A função date( ) espera que o argumento que você passa seja uma string de formato, representando o estilo de saída que você gostaria de obter. Cada uma das letras na string representa uma parte da data e da hora. H é a hora em um formato de 24 horas, i representa os minutos com um zero à esquerda onde for exigido, j é o dia do mês sem zero à esquerda, S representa o sufixo ordinal (neste caso th) e F é o nome completo do mês.

(Para uma lista completa de formatos suportados por date( ), veja o Capítulo 20.)

Acessando variáveis de formulário

A questão toda de utilizar o formulário de pedido é coletar o pedido do cliente. A obtenção de detalhes do que o cliente digitou é muito fácil no PHP, mas o método exato depende da versão do PHP que você estiver utilizando e de uma configuração em seu arquivo `php.ini`.

Variáveis do formulário

Dentro do script do PHP, você pode acessar cada um dos campos de formulário como uma variável do PHP cujo nome referencia o nome do campo de formulário. Você pode reconhecer nomes variáveis em PHP porque todos eles iniciam com um sinal de cifrão ($). (Esquecer o sinal de cifrão é um erro de programação comum.)

Dependendo da versão e configuração do PHP, há três maneiras de acessar os dados do formulário via variáveis. Esses métodos não têm nomes oficiais, então atribuímos a eles os nomes *abreviado*, *médio* e *longo*. Em qualquer caso, cada campo de formulário em uma página enviada a um script do PHP está disponível no script.

Você pode acessar o conteúdo do campo `tireqty` das seguintes maneiras:

```
$tireqty                        // estilo abreviado
$_POST['tireqty']               // estilo médio
$HTTP_POST_VARS['tireqty']      // estilo longo
```

Nesse exemplo e por todo este livro, utilizamos o estilo médio (`$_POST['tireqty']`) para referenciar variáveis de formulário, mas criamos versões abreviadas das variáveis para facilitar o uso. (Essa é a abordagem recomendada desde a versão PHP 4.2.0.)

Para seu próprio código, você poderia decidir utilizar uma abordagem diferente. Para tomar uma decisão informada, observe os diferentes métodos.

- O estilo abreviado (`$tireqty`) é conveniente, mas requer que a configuração de `register_globals` esteja ativada. Isso depende do padrão de cada versão do PHP. Em todas as versões, desde a 4.2.0, ela está desativada, por default. Anteriormente, ficava ativada, e a maioria dos programadores de PHP utilizava o estilo abreviado de tag. Essa mudança gerou muita confusão quando foi feita. Esse estilo também permite que você cometa erros que poderiam tornar seu código inseguro, e é por esse motivo que não é mais a abordagem recomendada.
- O estilo médio (`$_POST['tireqty']`) agora é a abordagem recomendada. É bem mais conveniente, mas começou apenas com o PHP 4.1.0, então não funciona em instalações antigas.
- O estilo longo (`$HTTP_POST_VARS['tireqty']`) é o mais prolixo. Mas como ele é obsoleto, é provavel que seja removido a longo prazo. Esse estilo costumava ser o mais portável, mas agora pode ser desativado por meio da diretiva de configuração `register_long_arrays`, o que melhora o desempenho.

Quando utilizar o estilo abreviado, os nomes das variáveis no script serão os mesmos que os nomes dos campos de formulário no formulário de HTML. Você não precisa declarar as variáveis nem tomar qualquer ação para criar essas variáveis em seu script. Elas são passadas para o script, essencialmente como argumentos são passados para uma função. Se estiver utilizando esse estilo, você poderá utilizar uma variável como `$tireqty`. O campo `tireqty` no formulário cria a variável `$tireqty` no script de processamento.

Um acesso conveniente assim para variáveis é tentador, mas antes de simplesmente ativar `register_globals`, vale a pena considerar por que a equipe de desenvolvimento do PHP o configura como desativado.

Ter acesso direto a variáveis como essa é muito conveniente, mas permite que você cometa erros de programação que poderiam comprometer a segurança de seus scripts. Com variáveis de formulário automaticamente transformadas em variáveis globais como essa, não há nenhuma separa-

ção óbvia entre as variáveis que você criou e as variáveis não-confiáveis que vieram diretamente do usuário.

Se você não for cuidadoso para dar às suas próprias variáveis um valor inicial, os usuários de seus scripts poderão passar as variáveis e os valores como variáveis de formulário que serão misturados às suas próprias variáveis. Se você optar por utilizar o estilo abreviado conveniente para acessar variáveis, precisará ter o cuidado de atribuir um valor inicial a todas as suas próprias variáveis.

O estilo médio envolve a recuperação das variáveis de formulário de um dos arrays `$_POST`, `$_GET` e `$_REQUEST`. Um de seus arrays `$_GET` ou `$_POST` manterá os detalhes de todas as variáveis de formulário. Qual array é utilizado depende se o método utilizado para enviar o formulário foi `POST` ou `GET`, respectivamente. Além disso, todos os dados enviados via `POST` ou `GET` estarão disponíveis por meio de `$_REQUEST`.

Se o formulário foi enviado por meio do método `POST`, então os dados inseridos na caixa `tireqty` serão armazenados em `$_POST['tireqty']`. Se o formulário foi enviado por meio de `GET`, então os dados estarão em `$_GET['tireqty']`. Em qualquer um caso, os dados estarão disponíveis em `$_REQUEST['tireqty']`.

Esses arrays são alguns dos novos chamados superglobais. Visitaremos novamente os superglobais quando falarmos sobre o escopo variável.

Se você estiver utilizando uma versão mais antiga do PHP, talvez não tenha acesso a `$_POST` ou `$_GET`. Antes da versão 4.1.0, essas informações eram armazenadas em arrays chamados `$HTTP_POST_VARS` e `$HTTP_GET_VARS`. Nós os chamamos estilo longo. Esse estilo pode ser utilizado tanto com as versões novas como com as antigas do PHP, mas agora está obsoleto e pode não funcionar com todas as versões futuras. Não há nenhum equivalente de `$_REQUEST` nesse estilo.

Se estiver usando o estilo longo, você poderá acessar a resposta do usuário por `$HTTP_POST_VARS['tireqty']` ou `$HTTP_GET_VARS['tireqty']`.

Os exemplos neste livro foram testados com o PHP versão 5.0 e às vezes serão incompatíveis com versões mais antigas do PHP, anteriores à 4.1.0. Recomendamos que você utilize a versão atual, sempre que possível.

Vejamos um exemplo. Como os nomes de variável de estilo longo e médio são um pouco incômodos e contam com um tipo de variável conhecida como arrays, que não abordaremos adequadamente até o Capítulo 3, começaremos criando cópias mais fáceis de utilizar.

Para copiar o valor de uma variável em outra, use o operador de atribuição, que no PHP é um sinal de igual (=). A linha de código a seguir criará uma nova variável denominada `$tireqty` e copiará o conteúdo de `$HTTP_POST_VARS['tireqty']` para a nova variável:

```
$tireqty = $_POST['tireqty'];
```

Coloque o seguinte bloco de código no início do script de processamento. Todos os outros scripts neste livro que tratam dos dados de um formulário conterão um bloco semelhante no início. Como não produzirá nenhuma saída, não faz nenhuma diferença colocar isso acima ou abaixo de `<html>` e outras tags HTML que iniciam a página. Geralmente colocamos esse bloco bem no início do script para que seja fácil de encontrá-lo.

```
<?php
 // cria nomes de variável abreviados
 $tireqty = $_POST['tireqty'];
 $oilqty = $_POST['oilqty'];
 $sparkqty = $_POST['sparkqty'];
?>
```

Esse código cria três novas variáveis – `$tireqty`, `$oilqty` e `$sparkqty` – e as configura para conter os dados que foram enviados via método `POST` a partir do formulário.

Para fazer o script começar a mostrar algo, acrescente as linhas a seguir na parte inferior do script do PHP:

```
echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br />';
echo $oilqty.' bottles of oil<br />';
echo $sparkqty.' spark plugs<br />';
```

Até aqui, você ainda não verificou o conteúdo das variáveis para se certificar de que dados relevantes foram inseridos em cada campo de formulário. Tente digitar de propósito dados errados e observe o que acontece. Depois de ler o restante deste capítulo, você provavelmente vai querer acrescentar validação de dados a esse script.

Se você agora carregar esse arquivo no navegador, a saída de script deverá se assemelhar ao que é mostrado na Figura 1.4. Os valores reais, naturalmente, dependerão do que você digitou no formulário.

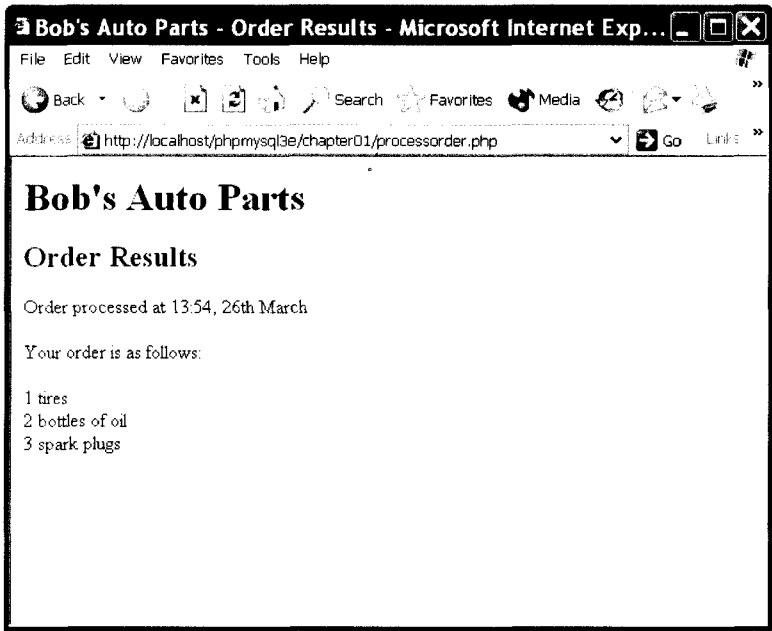



Figura 1.4 As variáveis de formulário digitadas pelo usuário são facilmente acessíveis no `processorder.php`.

As duas próximas subseções discutem dois elementos interessantes desse exemplo.

Concatenação de string

No script de exemplo, utilizamos `echo` para imprimir o valor que o usuário digitou em cada um dos campos de formulário, seguido por algum texto explanatório. Se observar atentamente as instruções `echo`, você verá que o nome da variável e o texto seguinte têm um ponto (.) entre eles, como neste exemplo:

```
echo $tireqty.' tires<br />';
```

Esse ponto é o operador de concatenação de string e é utilizado para adicionar strings (partes de texto). Você freqüentemente utilizará esse operador ao enviar saída para o navegador com `echo`. Isso é utilizado para evitar ter de escrever diversos comandos `echo`.

Para quaisquer variáveis não-array você também poderá colocar a variável dentro de uma string entre aspas duplas para ser ecoada. (Os arrays são um pouco mais complicados e então veremos a combinação de arrays e strings no Capítulo 4.)

Por exemplo:

```
echo "$tireqty tires<br />";
```

Essa é equivalente à primeira instrução. Qualquer formato é válido e o que você utilizar é uma questão de gosto pessoal. Note que isso é apenas um recurso de strings de aspas duplas. Você não pode colocar nomes de variáveis dentro de uma string entre aspas simples dessa forma. A execução da linha de código a seguir

```
echo '$tireqty tires<br />';
```

enviará `"$tireqty tires<br />"` para o navegador. Dentro de aspas duplas, o nome da variável será substituído por seu valor. Dentro de aspas simples, o nome da variável, ou qualquer outro texto, será enviado inalterado.

Variáveis e literais

A variável e a string que concatenamos em cada uma das instruções `echo` são tipos de coisas diferentes. Variáveis são um símbolo para dados. As strings são dados próprios. Quando utilizamos partes de dados brutos em um programa como esse, o chamamos literal para distingui-lo de uma variável. `$tireqty` é uma variável, um símbolo que representa os dados que o cliente digitou. Por outro lado, `' tires<br />'` é um literal. Ele pode ser considerado pelo seu valor nominal. Bem, quase. Lembra do segundo exemplo anteriormente? O PHP substituiu o nome variável `$tireqty` na string pelo valor armazenado na variável.

Lembre-se de que há dois tipos de strings no PHP – os valores com aspas duplas e os valores com aspas simples. O PHP tentará e avaliará strings entre aspas duplas, resultando no comportamento que vimos anteriormente. Strings entre aspas simples serão tratadas como verdadeiros literais.

Recentemente, foi acrescentada uma terceira maneira de especificar strings. A sintaxe heredoc (<<), familiar para os usuários de Perl, foi acrescentada ao PHP4. Ela permite especificar longas strings ordenadamente, especificando um identificador de fechamento que será usado para finalizar a string. O seguinte exemplo cria uma string de três linhas e a ecoa:

```
echo <<theEnd
 line 1
 line 2
 line 3
theEnd
```

O token `theEnd` é totalmente arbitrário. É preciso apenas garantir que ele não apareça no texto. Para fechar uma string heredoc, insira um identificador de fechamento no início de uma linha.

As strings heredoc são interpoladas, como strings delimitadas por aspas.

Entendendo identificadores

Os identificadores são os nomes de variáveis. (Os nomes de funções e classes também são identificadores – veremos funções e classes nos Capítulos 5 e 6.) Há algumas regras simples sobre identificadores:

- Os identificadores podem ser de qualquer comprimento e podem consistir em letras, números, sublinhados e sinais de cifrão.
- Os identificadores não podem iniciar com um dígito.
- No PHP, os identificadores diferenciam letras maiúsculas de minúsculas. `$tireqty` não é o mesmo que `$TireQty`. Tentar utilizar esses identificadores de modo intercambiável é um erro de programação comum. Os nomes de função são uma exceção a essa regra – seus nomes podem ser utilizados independente do uso de maiúsculas e minúsculas.
- Uma variável pode ter o mesmo nome que uma função. Mas isso é confuso e deve ser evitado. Além disso, você não pode criar uma função com o mesmo nome de outra.

Criando variáveis declaradas pelo usuário

Você pode declarar e utilizar suas próprias variáveis além das variáveis que lhe foram passadas a partir do formulário HTML.

Um dos recursos de PHP é que ele não requer que você declare variáveis antes de utilizá-las. Uma variável será criada quando você atribuir um valor a ela pela primeira vez – veja a próxima seção para detalhes.

Atribuindo valores a variáveis

Você atribui valores a variáveis utilizando o operador de atribuição, =, como fizemos ao copiar um valor de variável para outra. No site do Bob, queremos calcular o número total de itens encomendados e a quantidade total a pagar. Podemos criar duas variáveis para armazenar esses números. Em primeiro lugar, inicializaremos cada uma dessas variáveis para zero.

Adicione estas linhas à parte inferior do script PHP:

```
$totalqty = 0;
$totalamount = 0.00;
```

Cada uma dessas duas linhas cria uma variável e atribui um valor literal a ela. Você também pode atribuir valores variáveis às variáveis, por exemplo:

```
$totalqty = 0;
$totalamount = $totalqty;
```

Examinando os tipos de variável

O tipo de uma variável refere-se ao tipo de dados que é armazenado nela.

Tipos de dados do PHP

O PHP suporta os seguintes tipos de dados:

- **Integer** (Inteiro) – Utilizado para números inteiros.
- **Float** (também chamado **double**) (Dupla precisão) – Utilizado para números reais.
- **String** – Utilizado para strings de caracteres.
- **Boolean** – Utilizado para valores verdadeiros ou falsos.
- **Array** – Utilizado para armazenar vários itens de dados do mesmo tipo (veja o Capítulo 3).
- **Objeto** – Utilizado para armazenar instâncias de classes (veja o Capítulo 6).

Dois tipos extras também estão disponíveis – NULL e recurso. As variáveis que não receberam nenhum valor, nem foram zeradas ou receberam um valor específico `NULL` são do tipo NULL. Certas funções predefinidas retornam variáveis (como funções de banco de dados) que têm o tipo recurso. Elas representam recursos externos (tais como conexões de banco de dados). Você quase certamente não manipulará diretamente uma variável de recurso, mas freqüentemente ela é retornada por uma função e deve ser passada como parâmetro para outras funções.

Força do tipo

O PHP é uma linguagem de tipificação muito fraca. Na maioria das linguagens de programação, as variáveis só podem armazenar um tipo de dados e esse tipo deve ser declarado antes de a variável poder ser utilizada, como em C. No PHP, o tipo de uma variável é determinado pelo valor atribuído a ela.

Por exemplo, quando criamos $totalqty e $totalamount, seus tipos iniciais foram determinados, da seguinte maneira:

```
$totalqty = 0;
$totalamount = 0.00;
```

Como atribuímos 0, um inteiro, a $totalqty, agora ela é uma variável do tipo inteiro. De maneira semelhante, $totalamount é agora do tipo float.

Curiosamente, agora poderíamos adicionar uma linha ao script desse modo:

```
$totalamount = 'Hello';
```

A variável $totalamount então seria do tipo string. O PHP altera o tipo de variável de acordo com o que está armazenado nela em qualquer dado momento.

Essa capacidade de alterar tipos de maneira transparente e instantânea pode ser extremamente útil. Lembre-se de que o PHP sabe "automagicamente" que tipo de dados você coloca na variável. Ele retornará os dados com o mesmo tipo de dados quando você os recuperar a partir da variável.

Coerção de tipo

Você pode fingir que uma variável ou valor é de um tipo diferente utilizando uma coerção de tipo. Essas variáveis funcionam de modo idêntico ao modo como funcionam em C. Você simplesmente coloca o tipo temporário entre colchetes na frente da variável a que deseja aplicar a coerção.

Por exemplo, poderíamos ter declarado as duas variáveis anteriores utilizando uma coerção.

```
$totalqty = 0;
$totalamount = (double)$totalqty;
```

A segunda linha significa "aceite o valor armazenado em $totalqty, interprete-o como um número de dupla precisão e armazene-o em $totalamount". A variável $totalamount será do tipo float. A variável original em si não muda de tipo, então $totalqty permanece sendo do tipo inteiro.

Variáveis variáveis

O PHP fornece um outro tipo de variável – a variável variável. Essas variáveis permitem alterar dinamicamente o nome de uma variável.

Como você pode ver, o PHP permite muita liberdade nessa área – todas as linguagens permitirão alterar o valor de uma variável, mas muitas não permitirão alterar o tipo da variável e menos ainda permitirão alterar o nome da variável.

A maneira como isso funciona é utilizando o valor de uma variável como o nome de outra. Por exemplo, poderíamos configurar:

```
$varname = 'tireqty';
```

Podemos então usar $$varname no lugar de $tireqty. Por exemplo, podemos configurar o valor de $tireqty:

```
$$varname = 5;
```

Isso é exatamente equivalente a:

```
$tireqty = 5;
```

Isso talvez pareça um pouco obscuro, mas vamos rever sua utilização mais adiante. Em vez de ter de listar e utilizar cada variável de formulário separadamente, podemos utilizar um loop e uma variável para processar todas elas automaticamente. Há um exemplo que ilustra isso na seção sobre loops for.

Declarando e usando constantes

Como você viu anteriormente, podemos alterar o valor armazenado em uma variável. Também podemos declarar constantes. Uma constante armazena um valor como uma variável, mas seu valor é configurado uma vez e então não pode ser alterado em outra parte no script.

Em nossa aplicação de exemplo, poderíamos armazenar os preços para cada um dos itens na venda como constantes. Você pode definir essas constantes utilizando a função `define`:

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

Adicione essas linhas de código ao script. Agora, você tem três constantes que podem ser usadas para calcular o total do pedido do cliente.

Observe que os nomes da constante estão em letras maiúsculas. Essa é uma convenção emprestada do C que facilita distinguir rapidamente entre variáveis e constantes. Essa convenção não é necessária mas facilitará a leitura e a manutenção do código.

Uma diferença importante entre constantes e variáveis é que quando você referencia uma constante, ela não tem um sinal de cifrão na frente. Se quiser utilizar o valor de uma constante, utilize somente seu nome. Por exemplo, para utilizar uma das constantes que acabamos de criar, poderíamos digitar:

```
echo TIREPRICE;
```

Da mesma maneira como você define suas próprias constantes, o PHP configura as dele. Uma maneira fácil de obter uma visão geral dessas constantes é executar o comando `phpinfo( )`:

```
phpinfo( );
```

Isso fornecerá uma lista de constantes e variáveis predefinidas do PHP, entre outras informações úteis. Discutiremos algumas dessas à medida que prosseguirmos.

Outra diferença entre variáveis e constantes é que as constantes podem armazenar apenas dados boolean, integer, float ou string. Esses tipos são conhecidos como valores escalares.

Entendendo escopo de variável

O termo *escopo* refere-se aos lugares dentro de um script nos quais uma variável particular é visível. Os quatro tipos de escopo no PHP são como segue:

- As variáveis superglobais predefinidas são visíveis por toda parte dentro de um script.
- Constantes, uma vez declaradas, são sempre visíveis globalmente, ou seja, podem ser usadas dentro e fora das funções.
- Variáveis globais declaradas em um script são visíveis por todo esse script, mas *não dentro de funções.*
- Variáveis utilizadas dentro de funções que são declaradas como globais referem-se à variável global do mesmo nome.
- Variáveis criadas dentro de funções e declaradas como estáticas são invisíveis de fora da função, mas mantêm seu valor entre uma execução e outra da função. (Explicaremos essa idéia em detalhes no Capítulo 5.)
- Variáveis criadas dentro de funções são locais à função e deixam de existir quando a função é finalizada.

Do PHP 4.1 em diante, os arrays `$_GET` e `$_POST` e algumas outras variáveis especiais têm suas próprias regras de escopo. Essas são conhecidas como superglobais e podem ser vistas em toda par-

te, dentro e fora das funções.

A lista completa de superglobais é:

- $GLOBALS, um array de todas as variáveis globais.
- $_SERVER, um array de variáveis de ambiente do servidor.
- $_GET, um array de variáveis passadas ao script via método GET.
- $_POST, um array de variáveis passado ao script via método POST.
- $_COOKIE, um array de variáveis de cookie.
- $_FILES, um array de variáveis relacionadas a uploads de arquivo.
- $_ENV, um array de variáveis de ambiente.
- $_REQUEST, um array de todas as entradas do usuário, incluindo o conteúdo da entrada que tenha $_GET, $_POST, e $_COOKIE .
- $_SESSION, um array de variáveis de sessão.

Voltaremos a cada uma dessas durante todo o livro conforme seja relevante.

Abordaremos escopos em mais detalhe quando discutirmos as funções. Agora, todas as variáveis que utilizarmos serão globais por padrão.

Usando operadores

Os operadores são símbolos que podem ser utilizados para manipular valores e variáveis realizando uma operação neles. Precisaremos utilizar algum desses operadores para elaborar os totais e o imposto sobre o pedido do cliente.

Já mencionamos dois operadores: o operador de atribuição, (=), e (.), o operador de concatenação de string. Agora veremos a lista completa.

Em geral, os operadores podem aceitar um, dois ou três argumentos, com a maioria aceitando dois. Por exemplo, o operador de atribuição aceita dois – a posição na memória no lado esquerdo do símbolo = e uma expressão no lado direito. Esses argumentos são chamados de operandos, isto é, aquilo que está sendo operado.

Operadores aritméticos

Os operadores aritméticos são muito simples e diretos – eles são simplesmente os operadores matemáticos normais. Os operadores aritméticos são mostrados na Tabela 1.1.

Tabela 1.1 **Operadores aritméticos do PHP**

Operador	Nome	Exemplo
+	Adição	$a + $b
-	Subtração	$a - $b
*	Multiplicação	$a * $b
/	Divisão	$a / $b
%	Módulo	$a % $b

Com cada um desses operadores, podemos armazenar o resultado da operação. Por exemplo,

```
$result = $a + $b;
```

A adição e a subtração funcionam da maneira esperada. O resultado desses operadores é adicionar ou subtrair, respectivamente, os valores armazenados nas variáveis `$a` e `$b`.

Você também pode utilizar o símbolo de subtração, -, como um operador unário (isto é, um operador que aceita um argumento ou operando) para indicar números negativos.

```
$a = -1;
```

A multiplicação e a divisão também funcionam da maneira esperada. Observe o uso do asterisco como o operador de multiplicação, em vez do símbolo de multiplicação convencional e a barra normal como o operador de divisão, em vez do símbolo de divisão convencional.

O operador módulo retorna o resto da divisão da variável `$a` pela variável `$b`. Considere este fragmento de código:

```
$a = 27;
$b = 10;
$result = $a%$b;
```

O valor armazenado na variável `$result` é o resto quando dividimos 27 por 10; isto é, 7.

Você deve notar que os operadores aritméticos são normalmente aplicados a números inteiros ou de dupla precisão. Se você os aplicar a strings, o PHP tentará e converterá a string em um número. Se contiver um "e" ou um "E", ele será convertido em um número de dupla precisão; caso contrário, será convertido em um inteiro. O PHP procura dígitos no início da string e utiliza esses como o valor – se não houver nenhum, o valor da string será zero.

Operadores de string

Já vimos e utilizamos apenas o operador de string. Você pode utilizar o operador de concatenação de string para juntar duas strings e gerar e armazenar um resultado da mesma maneira como utilizaria o operador de adição para somar dois números.

```
$a = "Bob's ";
$b = 'Auto Parts';
$result = $a.$b;
```

A variável `$result` agora conterá a string `"Bob's Auto Parts"`.

Operadores de atribuição

Já vimos =, o operador de atribuição básico. Sempre se refira a esse como o operador de atribuição e o leia como "é configurado ou definido como". Por exemplo:

```
$totalqty = 0;
```

Isso deve ser lido como "`$totalqty` é configurado como zero". Explicaremos a razão disso quando discutirmos os operadores de comparação mais adiante neste capítulo.

Retornando valores de atribuição

Utilizar o operador de atribuição retorna um valor total semelhante aos outros operadores. Se você escrever:

```
$a + $b
```

O valor dessa expressão é o resultado da soma das variáveis `$a` e `$b`. De maneira semelhante, você pode escrever:

```
$a = 0;
```

O valor dessa expressão inteira é zero. Isso permite fazer operações como:

```
$b = 6 + ($a = 5);
```

Isso configurará o valor da variável `$b` como 11. Em geral, isso é verdadeiro sobre as atribuições: o valor da instrução de atribuição inteira é o valor que é atribuído ao operando do lado esquerdo.

Quando você elabora o valor de uma expressão, os parênteses podem ser utilizados para aumentar a precedência de uma subexpressão como fizemos aqui. Isso funciona exatamente da mesma maneira que em matemática.

Operadores de atribuição de combinação

Além da atribuição simples, há um conjunto de operadores de atribuição combinados. Cada um desses operadores é um modo abreviado de fazer outra operação com uma variável e atribuir o resultado de volta a essa variável. Por exemplo:

```
$a += 5;
```

Isso é equivalente a escrever:

```
$a = $a + 5;
```

Existem operadores de atribuição combinados para cada um dos operadores aritméticos e para o operador de concatenação de string. Um resumo de todos os operadores de atribuição combinados e seus efeitos é mostrado na Tabela 1.2.

Tabela 1.2 **Operadores de atribuição combinados do PHP**

Operador	Utilização	Equivalente a
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %= $b	$a = $a % $b
.=	$a .= $b	$a = $a . $b

Pré- e pós-incremento e decremento

Os operadores de pré- e pós- incremento (++) e decremento (--) são semelhantes aos operadores += e -=, mas com duas variações.

Todos os operadores de incremento têm dois efeitos: eles incrementam e atribuem um valor. Considere o seguinte:

```
$a=4;
echo ++$a;
```

A segunda linha utiliza o operador de pré-incremento, chamado assim porque o ++ aparece antes do `$a`. Isso tem o efeito de primeiro incrementar `$a` por 1, e segundo, retornar o valor incrementado. Nesse caso, `$a` é incrementado para 5 e então o valor 5 é retornado e impresso. O valor dessa expressão inteira é 5. (Note que o valor real armazenado em `$a` é alterado: não estamos apenas retornando `$a + 1`.)

Entretanto, se o ++ estiver depois do `$a`, estamos utilizando o operador de pós-incremento. Isso tem um efeito diferente. Considere o seguinte:

```
$a=4;
echo $a++;
```

Nesse caso, os efeitos são invertidos. Isto é, primeiro, o valor de $a é retornado e impresso e, segundo, é incrementado. O valor dessa expressão inteira é 4. Esse é o valor que será impresso. Mas o valor de $a depois que essa instrução for executada é 5.

Como você provavelmente pode adivinhar, o comportamento é semelhante ao do operador --. Entretanto, o valor de $a é decrementado em vez de incrementado.

Referências

O operador de referência, & ("E" comercial), pode ser utilizado em conjunção com a atribuição. Normalmente quando uma variável é atribuída à outra, é feita uma cópia da primeira variável e essa cópia é armazenada em outra parte na memória. Por exemplo:

```
$a = 5;
$b = $a;
```

Essas linhas de código fazem uma segunda cópia do valor em $a e a armazenam em $b. Se, subseqüentemente, alterarmos o valor de $a, $b não será alterado:

```
$a = 7; // $b será 5
```

Você pode evitar fazer uma cópia utilizando o operador de referência, &. Por exemplo:

```
$a = 5;
$b = &$a;
$a = 7; // $a e $b são agora 7
```

As referências podem ser um pouco complicadas. Lembre-se de que uma referência é como um alias, em vez de um ponteiro. Tanto $a como $b apontam para a mesma parte da memória. Você pode mudar isso desconfigurando uma delas, da seguinte forma:

```
unset($a);
```

A desconfiguração não muda o valor de $b (7), mas interrompe o vínculo entre $a e o valor 7 armazenado na memória.

Operadores de comparação

Os operadores de comparação são utilizados para comparar dois valores. As expressões que utilizam esses operadores retornam um valor lógico, true ou false, dependendo do resultado da comparação.

Operador de igualdade

O operador de comparação de igualdade, == (dois sinais de igual) permite testar se dois valores são iguais. Por exemplo, poderíamos utilizar a expressão

```
$a == $b
```

para testar se os valores armazenados em $a e $b são os mesmos. O resultado retornado por essa expressão será true se eles forem iguais ou false se não forem iguais.

É fácil confundir isso com =, o operador de atribuição. Isso funcionará sem causar um erro, mas geralmente não fornecerá o resultado desejado. De modo geral, valores que não sejam zero são avaliados como true, e valores zero, como false. Digamos que você inicializou duas variáveis da seguinte maneira:

```
$a = 5;
$b = 7;
```

Se você testar `$a = $b`, o resultado será verdadeiro. Por quê? O valor de `$a = $b` é o valor atribuído ao lado esquerdo, que nesse caso é 7. Esse é um valor não-zero, então a expressão é avaliada como `true`. Se pretendia testar `$a == $b`, que é avaliado como `false`, você introduziu um erro de lógica no código que pode ser extremamente difícil de localizar. Sempre verifique a utilização desses dois operadores e certifique-se de utilizar o operador que pretendia.

Esse é um erro fácil de fazer e você provavelmente o cometerá muitas vezes na sua carreira de programação.

Outros operadores de comparação

O PHP também suporta outros operadores de comparação. Um resumo de todos os operadores de comparação é mostrado na Tabela 1.3.

Um operador que deve ser notado é o operador de identidade (===), que retorna `true` apenas se os dois operandos foram iguais e do mesmo tipo. Por exemplo, `0=='0'` será verdadeiro, mas `0==='0'` não será, porque um zero é um inteiro e o outro é uma string.

Tabela 1.3 **Operadores de comparação do PHP**

Operador	Nome	Utilização
`==`	igual a	`$a == $b`
`===`	idêntico a	`$a === $b`
`!=`	não igual	`$a != $b`
`!==`	não idêntico	`$a !== $b`
`< >`	não igual (operador de comparação)	`$a < > $b`
`<`	menor que	`$a < $b`
`>`	maior que (operador de comparação)	`$a > $b`
`<=`	menor ou igual a	`$a <= $b`
`>=`	maior ou igual a	`$a >= $b`

Operadores lógicos

Os operadores lógicos são utilizados para combinar os resultados de condições lógicas. Por exemplo, poderíamos estar interessados em um caso em que o valor de uma variável, `$a`, está entre 0 e 100. Precisaríamos testar as condições `$a >= 0` e `$a <= 100`, utilizando o operador AND, assim:

```
$a >= 0 && $a <=100
```

O PHP suporta os operadores lógicos AND, OR, XOR (OR exclusivo) e NOT.

O conjunto de operadores lógicos e sua utilização estão resumidos na Tabela 1.4.

Tabela 1.4 **Operadores lógicos do PHP**

Operador	Nome	Utilização	Resultado
`!`	NOT	`!$b`	Retorna `true` se `$b` for `false` e vice-versa.
`&&`	AND	`$a && $b`	Retorna `true` se `$a` e `$b` forem `true`; caso contrário, `false`.
`||`	OR	`$a || $b`	Retorna `true` se `$a` ou `$b` ou ambos forem `true`; caso contrário, `false`.
`and`	AND	`$a and $b`	O mesmo que `&&`, mas com precedência mais baixa.
`or`	OR	`$a or $b`	O mesmo que `||`, mas com precedência mais baixa.

O mesmo que &&, mas com precedência mais baixa

Os operadores and e or têm precedência mais baixa que os operadores && e ||. Abordaremos a precedência detalhadamente mais adiante neste capítulo.

Operadores de bit a bit

Os operadores de bit a bit permitem tratar um inteiro como a série de bits utilizados para representá-lo.

Você provavelmente não encontrará grande utilidade para esses operadores no PHP, mas um resumo dos operadores de bit a bit é mostrado na Tabela 1.5.

Tabela 1.5 **Operadores de bit a bit do PHP**

<table>
<tr><th>Operador</th><th>Nome</th><th>Utilização</th><th>Resultado</th></tr>
<tr><td>&</td><td>AND bit a bit</td><td>$a & $b</td><td>Bits configurados em $a e $b são configurados no resultado.</td></tr>
<tr><td>|</td><td>OR bit a bit</td><td>$a | $b</td><td>Bits configurados em $a ou $b são configurados no resultado.</td></tr>
<tr><td>~</td><td>NOT bit a bit</td><td>~$a</td><td>Bits configurados em $a não são configurados no resultado e vice-versa.</td></tr>
<tr><td>^</td><td>XOR bit a bit</td><td>$a ^ $b</td><td>Bits configurados em $a ou $b mas não em ambos são configurados no resultado.</td></tr>
<tr><td><<</td><td>deslocamento para a esquerda</td><td>$a << $b</td><td>Desloca $a para a esquerda por $b bits.</td></tr>
<tr><td>>></td><td>deslocamento para a direita</td><td>$a >> $b</td><td>Desloca $a para a direita por $b bits.</td></tr>
</table>

Outros operadores

Além dos operadores que abordamos até agora, há vários outros.

O operador vírgula (,) é utilizado para separar argumentos de função e outras listas de itens. Em geral, ele é utilizado apenas casualmente.

Dois operadores especiais, new e ->, são utilizados para instanciar uma classe e acessar membros de classe, respectivamente. Esses serão abordados em detalhe no Capítulo 6.

Há três outros que discutiremos brevemente aqui.

O operador ternário

Esse operador (?:) funciona da mesma maneira como funciona em C. Ele tem a forma:

```
condição ? valor_se_true : valor_se_false
```

O operador ternário é semelhante à versão de expressão de uma instrução if-else, que é abordada mais adiante neste capítulo.

Um exemplo simples é:

```
($grade > 50 ? 'Passed' : 'Failed');
```

Essa expressão avalia notas de aluno como "Passed" ("Aprovado") ou "Failed" ("Reprovado").

O operador de supressão de erro

O operador de supressão de erro (@) poder ser utilizado na frente de qualquer expressão, isto é, qualquer coisa que gere ou tenha um valor. Por exemplo:

```
$a = @(57/0);
```

Sem o operador @, essa linha geraria um aviso de divisão por zero (experimente). Com o operador incluído, o erro é suprimido.

Se suprimir os avisos dessa maneira, você deve escrever algum código de tratamento de erro para verificar quando um aviso ocorreu. Se você configurar o PHP com o recurso track_errors ativado, a mensagem de erro será armazenada na variável global $php_errormsg.

O operador de execução

O operador de execução, na realidade, são dois operadores: na verdade, dois caracteres de acento grave (\` \`). O caractere de acento grave não é um caractere de aspas simples – ele geralmente está localizado na mesma tecla que o ~ (til).

O PHP tentará executar o que estiver entre os caracteres de acento grave como um comando na linha de comando do servidor. O valor da expressão é a saída do comando.

Por exemplo, sob sistemas operacionais do tipo UNIX, você pode utilizar:

```
$out = `ls -la`;
echo '<pre>'.$out.'</pre>';
```

Ou, equivalentemente em um servidor Windows:

```
$out = `dir c:`;
echo '<pre>'.$out.'</pre>';
```

Qualquer uma dessas versões obterá uma listagem de diretório e a armazenará em $out. Essa listagem, então, poderá ser ecoada para o navegador ou tratada de qualquer outra maneira.

Há outras maneiras de executar comandos no servidor. Abordaremos essas maneiras no Capítulo 18.

Operadores de array

Existem diversos operadores de array. Os operadores de elemento do array ([ ]) permitem acessar elementos do array. Você também pode utilizar o operador => em alguns contextos de array. Os operadores serão vistos no Capítulo 3.

Você também tem acesso a diversos outros operadores de array. Eles serão tratados em detalhes no Capítulo 3, mas os incluímos aqui por questões de clareza.

Tabela 1.6 **Operadores de array do PHP**

Operador	Nome	Utilização	Resultado
+	União	$a + $b	Retorna um array contendo tudo em $a e $b.
==	Igualdade	$a == $b	Retorna true se $a e $b tiverem os mesmos elementos.
===	Identidade	$a === $b	Retorna true se $a e $b tiverem os mesmos elementos na mesma ordem.
!=	Desigualdade	$a != $b	Retorna true se $a e $b não forem iguais.
<>	Desigualdade	$a <> $b	Retorna true se $a e $b não forem iguais.
!==	Não-identidade	$a !== $b	Retorna true se $a e $b não forem idênticos.

Você observará que todos os operadores de array da Tabela 1.6 possuem operadores equivalentes que funcionam em variáveis escalares. Contanto que você se lembre que + realiza adição em tipos escalares e união em arrays – mesmo que você não tenha interesse na aritmética por trás do comportamento – os comportamentos devem fazer sentido. Não é possível comparar arrays com tipos escalares.

O operador de tipo

Existe um operador de tipo, instanceof, usado em programação orientada a objetos, mas nós o mencionamos aqui para sermos completos. (A programação orientada a objetos será abordada no Capítulo 6.)

O operador instanceof permite verificar se um objeto é uma instância de uma classe específica, como neste exemplo:

```
class sampleClass{ };
$myObject = new sampleClass( );
if ($myObject instanceof sampleClass)
 echo "myObject is an instance of sampleClass";
```

Utilizando operadores: calculando os totais do formulário

Agora que você sabe utilizar os operadores PHP, está pronto para calcular os totais e o imposto no formulário de pedido de Bob. Para fazer isso, adicione o seguinte código à parte inferior do script de PHP:

```
$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br />';

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;

echo 'Subtotal: $'.number_format($totalamount,3).'<br />';

$taxrate = 0.10;  // o imposto de vendas local é 10%
$totalamount = $totalamount * (1 + $taxrate);
echo 'Total including tax: $'.number_format($totalamount,2).'<br />';
```

Se atualizar a página na janela de navegador, você deve ver uma saída semelhante à Figura 1.5.

Como você pode ver, temos vários operadores utilizados nessa parte do código. Utilizamos os operadores de adição (+) e multiplicação (*) para calcular as quantidades e o operador de concatenação de string (.) a fim de configurar a saída para o navegador.

Também utilizamos a função number_format( ) para formatar os totais como strings com duas casas decimais. Essa é uma função da biblioteca Math do PHP.

Se observar atentamente os cálculos, você poderia perguntar por que os cálculos foram feitos nessa ordem. Por exemplo, considere esta instrução:

```
$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;
```

A quantidade total parece estar correta, mas por que as multiplicações foram realizadas antes das adições? A resposta reside na *precedência* dos operadores, isto é, a ordem em que eles são avaliados.

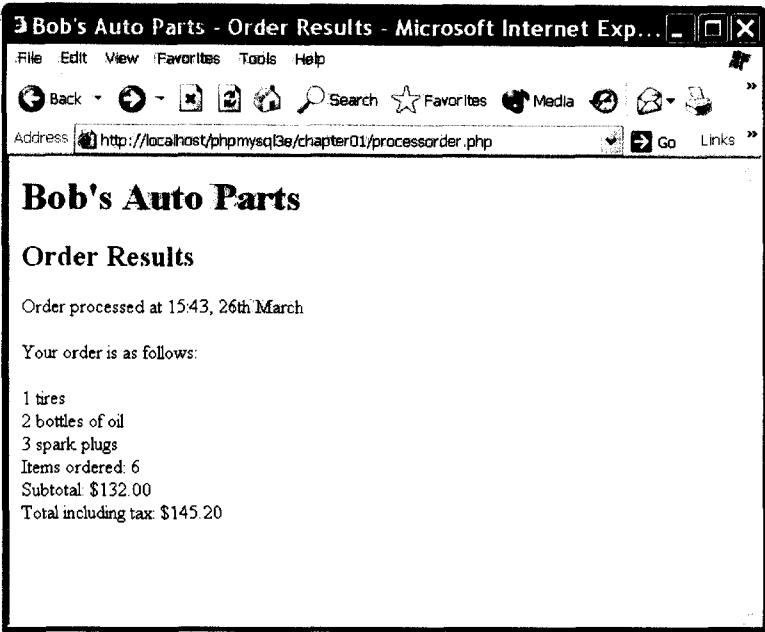



Figura 1.5 Os totais do pedido do cliente foram calculados, formatados e exibidos.

Entendendo a precedência e a associatividade: avaliando expressões

Em geral, os operadores têm uma precedência, ou ordem, configurada, em que eles são avaliados.

Os operadores também têm uma associatividade, que é a ordem em que os operadores da mesma precedência serão avaliados. Em geral, é da esquerda para a direita (chamada de esquerda para abreviar), e da direita para a esquerda (chamada de direita para abreviar) ou não-relevante.

A Tabela 1.7 mostra precedência de operadores e a associatividade em PHP.

Nessa tabela, os operadores de precedência mais baixos estão na parte superior e a precedência aumenta à medida que você desce pela tabela.

Tabela 1.7 **Precedência de operadores no PHP**

Associatividade	Operadores
Esquerda	,
Esquerda	`or`
Esquerda	`xor`
Esquerda	`and`
Direita	`print`
Esquerda	`= += -= *= /= .= %= &= |= ^= ~= <<= >>=`
Esquerda	`? :`
Esquerda	`||`
Esquerda	`&&`
Esquerda	`|`
Esquerda	`^`
Esquerda	`&`

Tabela 1.7 Continuação

Associatividade	Operadores
n/d	== != ===
n/d	< <= > >=
Esquerda	<< >>
Esquerda	+ - .
Esquerda	* / %
Direita	! ~ ++ -- (int) (double) (string) (array) (object) @
Direita	[ ]
n/d	new
n/d	( )

Observe que o operador de precedência mais alto é um que não discutimos ainda: os velhos e simples parênteses. O efeito desses é aumentar a precedência do que estiver contido dentro deles. Essa é a maneira como podemos contornar as regras de precedência quando precisamos.

Lembre-se desta parte do último exemplo:

```
$totalamount = $totalamount * (1 + $taxrate);
```

Se tivéssemos escrito

```
$totalamount = $totalamount * 1 + $taxrate;
```

o operador de multiplicação, tendo precedência mais alta que o operador de adição, seria realizado primeiro, fornecendo um resultado incorreto. Utilizando os parênteses, podemos forçar a subexpressão `1 + $taxrate` a ser avaliada primeiro.

Você pode utilizar quantos conjuntos de parênteses quiser em uma expressão. O conjunto de parênteses mais interno será avaliado primeiro.

Observe também outro operador nessa tabela que não vimos ainda: a construção da linguagem `print`, que é equivalente a `echo`. Ambas as construções geram saída.

Geralmente, utilizamos `echo` neste livro, mas você pode usar `print` se considerar mais legível. Nem `print` nem `echo` é realmente uma função, mas ambos podem ser chamados como uma função com os parâmetros entre parênteses. Eles também podem ser tratados como um operador: simplesmente insira a string após a palavra-chave `echo` ou `print`.

Chamar `print` como uma função faz com que ela retorne um valor `(1)`. Essa capacidade pode ser útil se você quiser gerar saída dentro de uma expressão mais complexa, mas não significa que `print` seja mais lento do que `echo`.

Usando funções de variável

Antes de deixarmos o mundo de variáveis e operadores, examinaremos as funções de variáveis do PHP. Essas são uma biblioteca de funções que permitem manipular e testar variáveis de maneiras diferentes.

Testando e configurando tipos de variável

A maioria dessas funções tem a ver com testar o tipo de uma função. As duas mais gerais são `gettype( )` e `settype( )`. Essas têm os seguintes protótipos de função; isto é, que argumentos elas esperam e o que eles retornam.

```
string gettype(mixed var);
bool settype (mixed var, string type);
```

Para utilizar gettype( ), passamos uma variável para ela. Ela determinará o tipo e retornará uma string contendo o nome de tipo:boolean, integer, double (para floats), string, array, object, resource ou NULL.Retorna unkown type se não for um dos tipos padrão.

Para utilizar settype( ), passamos para essa função uma variável cujo tipo gostaríamos de alterar e uma string contendo o novo tipo para essa variável, entre aqueles listados anteriormente.

Nota Este livro e a documentação php.net fazem referência ao tipo de dados misto. Esse tipo de dados não existe, mas como o PHP é tão flexível com manipulação de tipos, diversas funções podem ter muitos tipos de dados como argumento (ou qualquer um). Os argumentos para os quais muitos dados são permitidos são mostrados com o pseudotipo misto.

Podemos utilizar essas funções da seguinte maneira:

```
$a = 56;
echo gettype($a).'<br />';
settype($a, 'double');
echo gettype($a).'<br />';
```

Quando gettype( ) é chamado pela primeira vez, o tipo de $a é inteiro. Após a chamada a settype( ), o tipo será alterado para double.

O PHP também fornece algumas funções de teste de tipo específico. Cada uma dessas funções aceita uma variável como argumento e retorna true ou false. As funções são:

- is_array( )
- is_double( ), is_float( ), is_real( ) (Todas com a mesma função)
- is_long( ), is_int( ), is_integer( ) (Todas com a mesma função)
- is_string( )
- is_object( )
- is_resource( )
- is_null( )
- is_scalar( ) – Verifica se a variável é uma escalar, ou seja, integer, boolean, string ou float.
- is_numeric( ) – Verifica se a variável é algum tipo de número ou string numérica.
- is_callable( ) – Verifica se a variável é o nome de uma função válida.

Testando o status da variável

O PHP possui várias maneiras de testar o status de uma variável. A primeira dessas é isset( ), que tem o seguinte protótipo:

```
boolean isset(mixed var);
```

Essa função aceita um nome de variável como argumento e retorna true se ele existir e false caso contrário.

Você pode remover uma variável utilizando sua construção companheira, unset( ). Essa tem o seguinte protótipo:

```
void unset(mixed var);
```

Essa função elimina a variável que é passada.

A função empty( ) verifica se existe uma variável e se tem um valor não-zero e não-vazio e retorna true ou false de acordo com o resultado. Ela tem o seguinte protótipo:

```
boolean empty(mixed var);
```

Vejamos um exemplo que utiliza essas maneiras.

Tente adicionar o seguinte código a seu script temporariamente:

```
echo isset($tireqty);
echo isset($nothere);
echo empty($tireqty);
echo empty($nothere);
```

Atualize a página para ver os resultados.

A variável $tireqty deve retornar true a partir de isset( ) independente do valor que você inseriu ou não inseriu nesse campo de formulário. Se ela é ou não empty( ) depende do que você inseriu nela.

A variável $nothere não existe, então ela gerará um resultado false a partir de isset( ) e um resultado true a partir de empty( ).

Essas funções podem ser úteis para certificar-se de que o usuário preencheu os campos apropriados no formulário.

Reinterpretando variáveis

Você pode alcançar o equivalente a lançar uma variável chamando uma função. As três funções que podem ser úteis para isso são:

```
int intval(mixed var);
float doubleval(mixed var);
string strval(mixed var);
```

Cada uma aceita uma variável como entrada e retorna o valor da variável convertido para o tipo apropriado. A função intval( ) também permite especificar a base para a conversão quando a variável a ser convertida é uma string. (Dessa forma, você pode converter, por exemplo, strings hexadecimais em inteiros.)

Implementando estruturas de controle

As estruturas de controle são as estruturas dentro de uma linguagem que permitem controlar o fluxo de execução por meio de um programa ou script. Você pode agrupá-las em estruturas condicionais (ou de desvio) e estruturas de repetição ou loops. Consideraremos as implementações específicas de cada uma dessas no PHP em seguida.

Tomando decisões com condicionais

Se quisermos responder sensatamente à entrada do nosso usuário, nosso código precisa ser capaz de tomar decisões. As construções que instruem nosso programa a tomar decisões são chamadas condicionais.

Instruções `if`

Podemos utilizar uma instrução if para tomar uma decisão. Você deve fornecer à instrução if uma condição para utilização. Se a condição for true, o bloco de código seguinte será executado. As condições nas instruções if devem estar entre parênteses ( ).

Por exemplo, se recebermos um pedido sem nenhum produto selecionado, provavelmente é porque acidentalmente pressionamos o botão Submit. Em vez de informar "Pedido processado", a página poderia fornecer uma mensagem mais útil.

Quando o visitante não encomenda nenhum item, poderíamos querer dizer: "You did not order anything on the previous page!" (Você não selecionou nada na página anterior!) Podemos fazer isso facilmente com a seguinte instrução `if`:

```
if( $totalqty == 0 )
echo 'You did not order anything on the previous page!<br />';
```

A condição que estamos utilizando é `$totalqty = = 0`. Lembre-se de que o operador igual (==) comporta-se de forma diferente do operador de atribuição (=).

A condição `$totalqty == 0` será `true` se `$totalqty` for igual a zero. Se `$totalqty` não for igual a zero, a condição será `false`. Quando a condição for `true`, a instrução `echo` será executada.

Blocos de código

Freqüentemente, temos mais de uma instrução que queremos executar dentro de uma instrução condicional como `if`. Não há nenhuma necessidade de colocar uma nova instrução `if` antes de cada uma. Em vez disso, podemos agrupar um número de instruções como um bloco. Para declarar um bloco, coloque-o entre chaves:

```
if( $totalqty == 0 )
{
  echo '<font color=red>';
  echo 'You did not order anything on the previous page!<br />';
  echo '</font>';
}
```

As três linhas de código entre chaves são agora um bloco de código. Quando a condição for `true`, todas as três linhas serão executadas. Quando a condição for `false`, todas as três linhas serão ignoradas.

Nota Como já mencionado, o PHP não se importa com a maneira como você configura o código. Você deve recuar o código para fins de legibilidade. O recuo geralmente é utilizado para permitir uma visão geral das linhas que só serão executadas se condições forem atendidas, de quais instruções são agrupadas em blocos e de quais instruções fazem parte de loops ou funções. Você pode ver nos exemplos anteriores que a instrução que depende da instrução `if` e as instruções que compõem o bloco são recuadas.

Instruções `else`

Com freqüência, você precisará decidir não apenas se deseja realizar uma ação, mas também qual conjunto de possíveis ações deseja realizar.

Uma instrução `else` permite definir que uma ação alternativa seja tomada quando a condição em uma instrução `if` for `false`. Queremos avisar aos clientes de Bob se eles tentarem fechar o pedido sem selecionar nenhum produto. Por outro lado, se eles realmente preencherem um pedido, exibiremos o que eles encomendaram em vez de um aviso.

Se reorganizarmos nosso código e adicionarmos uma instrução `else`, podemos exibir um aviso ou um resumo.

```
if( $totalqty == 0 )
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  echo $tireqty.' tires<br />';
```

```
  echo $oilqty.' bottles of oil<br />';
  echo $sparkqty.' spark plugs<br />';
}
```

Podemos construir processos lógicos mais complexos aninhando as instruções `if` uma dentro da outra. No código a seguir, além de o resumo ser exibido somente se a condição `$totalqty == 0` for `true`, cada linha no resumo será exibida somente se sua própria condição for satisfeita.

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  if ( $tireqty>0 )
    echo $tireqty.' tires<br />';
  if ( $oilqty>0 )
    echo $oilqty.' bottles of oil<br />';
  if ( $sparkqty>0 )
    echo $sparkqty.' spark plugs<br />';
}
```

Instruções `elseif`

Para muitas das decisões que tomamos, há mais de duas opções. Podemos criar uma seqüência de muitas opções utilizando a instrução `elseif`. Essa instrução é uma combinação de uma instrução `else` e uma `if`. Fornecendo uma seqüência de condições, o programa pode verificar cada uma até encontrar uma que seja `true`.

Bob fornece um desconto para grandes pedidos de pneus. O esquema de desconto funciona assim:

- Menos de 10 pneus comprados – nenhum desconto;
- 10-49 pneus comprados – 5% desconto;
- 50-99 pneus comprados – 10% desconto;
- 100 ou mais pneus comprados – 15% desconto.

Podemos criar código para calcular o desconto utilizando condições e instruções `if` e `elseif`. Precisamos utilizar o operador AND (&&) para combinar duas condições em uma.

```
if( $tireqty < 10 )
  $discount = 0;
elseif( $tireqty >= 10 && $tireqty <= 49 )
  $discount = 5;
elseif( $tireqty >= 50 && $tireqty <= 99 )
  $discount = 10;
elseif( $tireqty >= 100 )
  $discount = 15;
```

Note que você está livre para digitar `elseif` ou `else if` – com ou sem espaço, ambos estão corretos.

Se pensar em escrever um conjunto de instruções `elseif` em cascata, você deve estar ciente de que somente um dos blocos ou instruções será executado. Isso não fez diferença nesse exemplo porque todas as condições eram mutuamente exclusivas – somente uma pode ser verdadeira de cada vez. Se escrevêssemos nossas condições de modo que mais de uma pudesse ser verdadeira ao mesmo tempo, somente o bloco ou instrução seguinte à primeira condição verdadeira seria executado.

Instruções switch

A instrução switch funciona de modo semelhante ao da instrução if, mas permite que a condição aceite mais de dois valores. Em uma instrução if, a condição pode ser true ou false. Em uma instrução switch, a condição pode aceitar qualquer número de valores diferentes, contanto que seja avaliada como um tipo simples (inteiro, string ou float). Você precisa fornecer uma instrução case a fim de tratar cada valor para o qual deseja tomar uma ação e, opcionalmente, um case padrão a fim de tratar qualquer outro valor para o qual você não forneceu uma instrução case específica.

Bob quer saber quais formas de publicidade estão funcionando para ele. Podemos adicionar uma pergunta ao nosso formulário de pedido.

Insira este HTML no formulário de pedido e o formulário ficará semelhante à Figura 1.6:

```
<tr>
  <td>How did you find Bob's</td>
  <td><select name="find">
        <option value = "a">I'm a regular customer
        <option value = "b">TV advertising
        <option value = "c">Phone directory
        <option value = "d">Word of mouth
      </select>
  </td>

</tr>
```

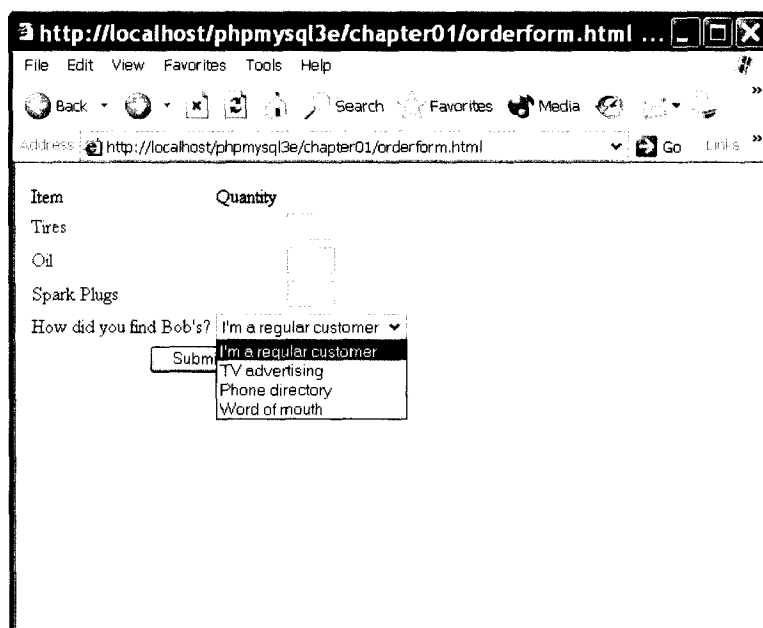



Figura 1.6 O formulário de pedido agora pergunta aos visitantes como eles encontraram Bob's Auto Parts.

Esse código HTML adicionou uma nova variável de formulário cujo valor será "a", "b", "c" ou "d". Poderíamos tratar essa nova variável com uma série de instruções if e elseif da seguinte maneira:

```
if($find == 'a')
  echo '<p>Regular customer.</p>';
elseif($find == 'b')
  echo '<p>Customer referred by TV advert.</p>';
elseif($find == 'c')
```

```
  echo '<p>Customer referred by phone directory.</p>';
elseif($find == 'd')
  echo '<p>Customer referred by word of mouth.</p>';
```

Alternativamente, poderíamos escrever uma instrução `switch`:

```
switch($find)
{
  case 'a' :
    echo '<p>Regular customer.</p>';
    break;
  case 'b' :
    echo '<p>Customer referred by TV advert.</p>';
    break;
  case 'c' :
    echo '<p>Customer referred by phone directory.</p>';
    break;
  case 'd' :
    echo '<p>Customer referred by word of mouth.</p>';
    break;
  default :
    echo '<p>We do not know how this customer found us.</p>';
    break;
}
```

(Observe que os dois exemplos assumem que você extraiu `$find` do array `$_POST`.)

A instrução `switch` comporta-se de maneira um pouco diferente de uma instrução `if` ou `elseif`. Uma instrução `if` só afeta uma instrução a menos que você deliberadamente utilize chaves para criar um bloco de instruções. Uma instrução `switch` comporta-se de maneira oposta. Quando um caso em uma instrução `switch` for ativado, o PHP executará instruções até alcançar uma instrução `break`. Sem instruções `break`, um `switch` executaria todo o código após o `case` que fosse verdadeiro. Quando uma instrução `break` for alcançada, a próxima linha de código depois da instrução `switch` será executada.

Comparando as diferentes condicionais

Se não conhece essas instruções, talvez você se pergunte, "qual é a melhor?"

Essa não é realmente uma pergunta a que podemos responder. Não há nada que você possa fazer com uma ou mais instruções `else`, `elseif` ou `switch` que não possa fazer com um conjunto de instruções `if`. Você deve tentar utilizar a condicional que for mais legível na sua situação. Você adquirirá uma noção melhor sobre isso com a experiência.

Repetindo ações com iterações

Uma coisa em que os computadores sempre foram muito bons é automatizar tarefas repetitivas. Se houver algo que você precisa preparar da mesma maneira muitas vezes, pode utilizar um loop para repetir algumas partes do programa.

O Bob quer uma tabela que exiba o custo do frete que será adicionado ao pedido de um cliente. Com o serviço de correio que o Bob utiliza, o custo de frete depende da distância do destinatário. O custo pode ser elaborado com uma fórmula simples.

Queremos que a nossa tabela de frete seja semelhante à tabela da Figura 1.7.

A Listagem 1.2 mostra a HTML que exibe essa tabela. Você pode ver que essa listagem é longa e repetitiva.

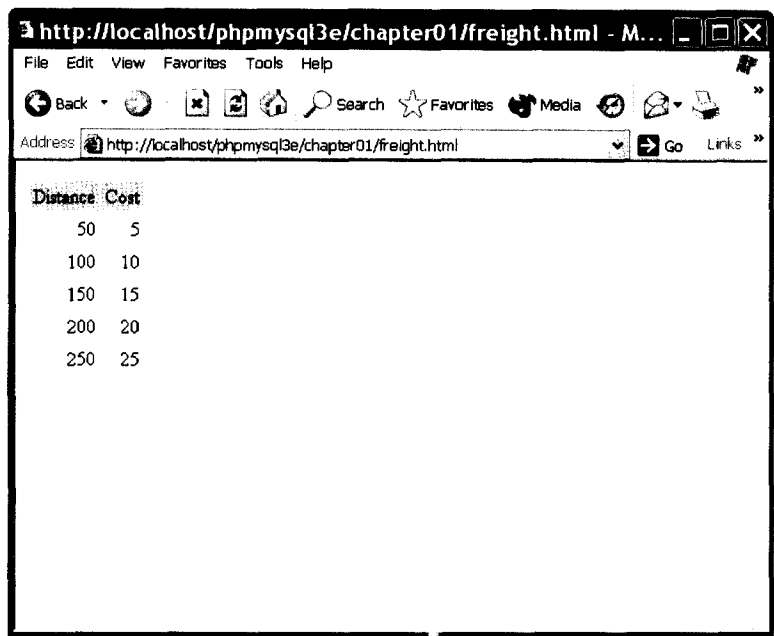



Figura 1.7 Essa tabela mostra o custo de frete de acordo com a distância.

Listagem 1.2 **freight.html** – HTML para a tabela de frete do Bob

```
<html>
<body>
<table border="0" cellpadding="3">
<tr>
  <td bgcolor="#CCCCCC" align="center">Distance</td>
  <td bgcolor="#CCCCCC" align="center">Cost</td>
</tr>
<tr>
  <td align="right">50</td>
  <td align="right">5</td>
</tr>
<tr>
  <td align="right">100</td>
  <td align="right">10</td>
</tr>
<tr>
  <td align="right">150</td>
  <td align="right">15</td>
</tr>
<tr>
  <td align="right">200</td>
  <td align="right">20</td>
</tr>
<tr>
  <td align="right">250</td>
  <td align="right">25</td>
</tr>
</table>
</body>
</html>
```

Seria útil se, em vez de requerer que um humano entediado – que deve ser pago por seu tempo – digitasse a HTML, um computador barato e incansável fizesse isso.

As instruções de loop instruem o PHP a executar uma instrução ou bloco repetidamente.

Loops while

O tipo mais simples de loop no PHP é o loop while. Como uma instrução if, ele conta com uma condição. A diferença entre um loop while e uma instrução if é que uma instrução if executa uma vez o bloco de código seguinte se a condição for true. Um loop while executa o bloco repetidamente contanto que a condição seja true.

Geralmente, você utiliza um loop while quando não sabe quantas iterações serão requeridas para tornar a condição verdadeira. Se você requer um número fixo de iterações, considere a utilização de um loop for.

A estrutura básica de um loop while é:

```
while( condição ) expressão;
```

O seguinte loop while exibirá os números de 1 a 5.

```
$num = 1;
while ($num <= 5 )
{
  echo $num."<br />";
  $num++;
}
```

No começo de cada iteração, a condição é testada. Se a condição for false, o bloco não será executado e o loop terminará. A próxima instrução após o loop será então executada.

Podemos utilizar um loop while para fazer algo mais útil, como exibir a tabela repetitiva de frete na Figura 1.7. A Listagem 1.3 utiliza um loop while para gerar a tabela de frete.

Listagem 1.3 **freight.php – Gerando a tabela de frete do Bob com o PHP**

```
<html>
<body>
<table border="0" cellpadding="3">
<tr>
 <td bgcolor="#CCCCCC" align="center">Distance</td>
 <td bgcolor="#CCCCCC" align="center">Cost</td>
</tr>
<?php
$distance = 50;
while ($distance <= 250 )
{
 echo "<tr>\n  <td align = 'right'>$distance</td>\n";
 echo "  <td align = 'right'>". $distance / 10 ."</td>\n</tr>\n";
 $distance += 50;
}
?>
</table>
</body>
</html>
```

A fim de tornar o HTML gerado por nosso script legível, é necessário incluir novas linhas e espaços. Como já mencionado, os navegadores ignorarão isso, mas é importante para leitores humanos. Você freqüentemente precisa ver na HTML se sua saída não for o que estava buscando.

Na Listagem 1.3, você verá \n dentro de algumas strings. Quando estiver dentro de uma string entre aspas duplas, essa seqüência de caracteres representará um novo caractere de linha.

Loops for e foreach

A maneira como utilizamos os loops while anteriormente é muito comum. Configuramos um contador com o qual iniciar. Antes de cada iteração, testamos o contador em uma condição. No final de cada iteração, modificamos o contador.

Podemos escrever esse estilo de loop de uma forma mais compacta utilizando um loop for.

A estrutura básica de um loop for é:

```
for( expressão1; condição; expressão2)
  expressão3;
```

- *expressão1* é executada uma vez no início. Aqui você normalmente configurará o valor inicial de um contador.
- A expressão *condição* é testada antes de cada iteração. Se a expressão retornar false, a iteração pára. Aqui, você normalmente testará o contador contra um limite.
- *expressão2* é executada no final de cada iteração. Aqui você normalmente ajustará o valor do contador.
- *expressão3* é executada uma vez por iteração. Essa expressão normalmente é um bloco de código e conterá o corpo principal do código de loop.

Podemos reescrever o exemplo de loop while na Listagem 1.3 como um loop for. O código de PHP se tornará:

```
<?php
for($distance = 50; $distance <= 250; $distance += 50)
{
  echo "<tr>\n  <td align='right'>$distance</td>\n";
  echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
}
?>
```

As versões while e for são funcionalmente idênticas. O loop for é um pouco mais compacto, economizando duas linhas.

Esses dois tipos de loop são equivalentes – nenhum é melhor ou pior que o outro. Em uma determinada situação, você pode utilizar aquele que achar mais intuitivo.

Como uma observação à parte, você pode combinar variáveis variáveis com um loop for para iterar por uma série de campos repetitivos de formulário. Se, por exemplo, você tiver campos de formulário com nomes como name1, name2, name3 e assim por diante, pode processá-los assim:

```
for ($i=1; $i <= $numnames; $i++)
{
  $temp= "name$i";
  echo $$temp.'<br />'; // ou qualquer que seja o processamento que deseje fazer
}
```

Criando dinamicamente os nomes das variáveis, podemos acessar cada um dos campos de cada vez.

Assim como o loop for, há um loop foreach projetado especificamente para utilização com arrays. Discutiremos como utilizá-lo no Capítulo 3.

Loops do..while

O tipo final de loop que mencionaremos se comporta de modo ligeiramente diferente. A estrutura geral de uma instrução do..while é:

```
do
  expressão;
while( condição );
```

Um loop do..while difere de um loop while porque a condição é testada no final. Isso significa que em um loop do..while, a instrução ou bloco dentro do loop sempre é executado pelo menos uma vez.

Mesmo se tomarmos o exemplo em que a condição é false no início e nunca se torna true, o loop será executado uma vez antes de verificar a condição e terminará.

```
$num = 100;
do
{
  echo $num.'<br />';
}
while ($num < 1 );
```

Interrompendo uma estrutura de controle ou script

Se você quiser parar de executar uma parte de código, há três abordagens, dependendo do efeito que estiver tentando alcançar.

Se quiser parar de executar um loop, você pode utilizar a instrução break como discutido anteriormente na seção sobre switch. Se utilizar a instrução break em um loop, a execução do script continuará na próxima linha do script após o loop.

Se quiser pular para a próxima iteração de loop, você pode utilizar, em vez disso, a instrução continue.

Se quiser terminar de executar o script de PHP inteiro, você pode utilizar exit. Em geral, isso é útil ao realizar verificação de erros. Por exemplo, poderíamos modificar nosso exemplo anterior da seguinte maneira:

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
  exit;
}
```

A chamada para exit interrompe a execução do resto do script de PHP.

Empregando sintaxe alternativa de estrutura de controle

Há uma forma alternativa de sintaxe para todas as estruturas de controle que vimos aqui. Ela consiste em substituir a chave de abertura ({) por um dois-pontos (:) e a chave de fechamento por uma nova palavra-chave, que será endif, endswitch, endwhile, endfor ou endforeach, dependendo de como a estrutura de controle está sendo utilizada. Não há sintaxe alternativa disponível para loops do...while.

Por exemplo, o código

```
 if( $totalqty == 0)
{
 echo 'You did not order anything on the previous page!<br />';
 exit;
}
```

poderia ser convertido para esta sintaxe alternativa utilizando as palavras-chave if e endif:

```
if( $totalqty == 0):
 echo 'You did not order anything on the previous page!<br />';
 exit;
endif;
```

Utilizando declare

Uma outra estrutura de controle no PHP, a estrutura declare, não é tão usada na codificação diária quanto as outras construções. O formato geral dessa estrutura de controle é:

```
declare (directive)
{
// bloqueio
}
```

Ela é utilizada para definir *diretivas de execução* para o bloco do código – ou seja, as regras sobre como o código seguinte deve ser executado. Atualmente, apenas uma diretiva de execução, chamada ticks, foi implementada. Para defini-la, insira a diretiva ticks=n. Ela permite executar uma função específica a cada n linhas de código dentro do bloco de código, o que é especialmente útil para análise de perfil e depuração.

A estrutura de controle declare é mencionada aqui apenas a título de ilustração. Veremos alguns exemplos mostrando como utilizar as funções tick nos Capítulos 24 e 25.

A seguir: salvando o pedido do cliente

Agora você sabe como receber e manipular o pedido do cliente. No próximo capítulo, veremos como armazenar o pedido para que ele possa ser recuperado e atendido mais tarde.

2

Armazenando e recuperando dados

AGORA QUE SABEMOS ACESSAR E MANIPULAR DADOS inseridos em um formulário HTML, podemos examinar maneiras de armazenar essas informações para posterior utilização. Na maioria dos casos, incluindo o exemplo visto no capítulo anterior, você deve querer armazenar esses dados e carregá-los mais tarde. No nosso caso, precisamos armazenar os pedidos de clientes de modo que eles possam ser atendidos mais tarde.

Neste capítulo, examinaremos como você pode gravar o pedido do cliente do exemplo anterior em um arquivo e ler outra vez. Também discutiremos por que nem sempre essa é uma boa solução. Quando temos muitos pedidos, devemos utilizar um sistema de gerenciamento de bancos de dados como MySQL.

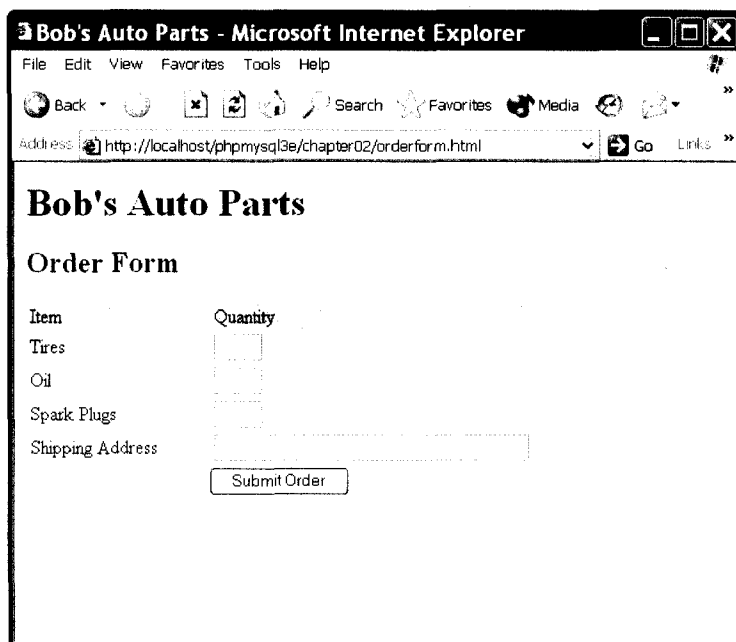
Os tópicos-chave que você aprenderá neste capítulo incluem:

- Salvar dados para mais tarde;
- Abrir um arquivo;
- Criar e gravar para um arquivo;
- Fechar um arquivo;
- Ler um arquivo;
- Bloquear arquivo;
- Excluir arquivos;
- Outras funções de arquivo úteis;
- Fazer de uma maneira melhor: sistemas de gerenciamento de bancos de dados.

Salvando dados para mais tarde

Há basicamente duas maneiras de armazenar dados: em arquivos simples ou em um banco de dados.

Um arquivo *flat file* pode ter muitos formatos mas, em geral, quando nos referimos a ele, queremos dizer um arquivo de texto simples. Nesse exemplo, gravaremos pedidos de clientes para um arquivo de texto, um pedido por linha.

Isso é muito simples de fazer, mas também muito restrito, como veremos mais adiante neste capítulo. Se estiver lidando com informações de qualquer volume razoável, você provavelmente deve querer utilizar um banco de dados em vez disso. Entretanto, arquivos simples têm suas utilizações e há algumas situações em que você precisará saber como utilizá-los.

Gravar arquivos no PHP e ler a partir deles é praticamente idêntico ao modo como isso é feito em C. Se você fez alguma programação C ou criação de script de shell UNIX, isso lhe será muito familiar.

Armazenando e recuperando pedidos do Bob

Neste capítulo, utilizaremos uma versão ligeiramente modificada do formulário de pedido que vimos no último capítulo. Iniciaremos com esse formulário e o código de PHP que escrevemos para processar os dados do pedido.

Nota Scripts de HTML e PHP utilizados neste capítulo podem ser encontrados na pasta `chapter2/` do CD-ROM deste livro.

Modificamos o formulário para incluir uma maneira rápida de obter o endereço de entrega do cliente. Você pode ver esse formulário na Figura 2.1.

Figura 2.1 Essa versão do formulário de pedido obtém o endereço de entrega do cliente.

O campo de formulário para o endereço de entrega é chamado `address`. Isso nos dá uma variável que podemos acessar como `$address` quando processamos o formulário no PHP, se tivermos `register_globals` ativado, ou como `$_POST['address']` ou `$_GET['address']`, se `register_globals` estiver desativado (veja o Capítulo 1, para mais detalhes).

Neste capítulo, gravaremos cada pedido que entrar no mesmo arquivo. Então, construiremos uma interface Web para o pessoal do Bob visualizar os pedidos que foram recebidos.

Visão geral do processamento de arquivo

Há três passos para gravar dados em um arquivo:

1. Abrir o arquivo. Se o arquivo não existe ainda, ele precisará ser criado.
2. Gravar os dados no arquivo.
3. Fechar o arquivo.

De maneira semelhante, há três passos para ler dados de um arquivo:

1. Abrir o arquivo. Se o arquivo não puder ser aberto (por exemplo, se não existir), precisamos reconhecer isso e sair elegantemente.

2. Ler dados a partir do arquivo.
3. Fechar o arquivo.

Quando quiser ler dados a partir de um arquivo, você tem escolhas sobre quanto do arquivo ler por vez. Veremos cada uma dessas escolhas em detalhe. Por enquanto, começaremos abrindo um arquivo.

Abrindo um arquivo

Para abrir um arquivo no PHP, utilizamos a função `fopen( )`. Quando abrimos o arquivo, precisamos especificar como pretendemos utilizá-lo. Isso é conhecido como *modo de arquivo*.

Escolhendo modos de arquivo

O sistema operacional no servidor precisa saber o que você quer fazer com um arquivo que esteja abrindo. O sistema precisa saber se o arquivo pode ser aberto por outro script enquanto você estiver com o arquivo aberto e descobrir se você (o proprietário do script) tem permissão para utilizá-lo dessa maneira. Essencialmente, os modos de arquivo fornecem um mecanismo para o sistema operacional determinar como tratar solicitações de acesso de outras pessoas ou scripts e um método para verificar se você tem acesso e permissão para esse arquivo particular.

Há três escolhas que você precisa fazer ao abrir um arquivo:

1. Talvez você queira abrir um arquivo somente para leitura, somente para gravação ou tanto para leitura como para gravação.
2. Se gravar em um arquivo, talvez você queira sobrescrever qualquer conteúdo existente de um arquivo ou acrescentar novos dados ao final do arquivo. Talvez você queira terminar seu programa de forma elegante em vez de sobrescrever um arquivo se ele já existir.
3. Se estiver tentando gravar em um arquivo em um sistema que diferencia entre arquivos de texto e binários, talvez você queira especificar isso.

A função `fopen( )` suporta combinações dessas três opções.

Utilizando **`fopen( )`** para abrir um arquivo

Vamos assumir que queremos gravar um pedido de cliente para o arquivo de pedido do Bob. Você pode abrir esse arquivo para gravar com o seguinte:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", "w");
```

Quando `fopen` é chamado, ele espera dois ou três parâmetros. Normalmente, você utilizará dois, como é mostrado nessa linha de código.

O primeiro parâmetro deve ser o arquivo que você deseja abrir. Você pode especificar um caminho para esse arquivo da maneira como fizemos no código anterior – nosso arquivo `orders.txt` está no diretório de pedidos. Utilizamos a variável predefinida `$HTTP_SERVER_VARS['DOCUMENT_ROOT']` do PHP, mas para os incômodos nomes completos de variáveis de formulário, atribuímos um nome mais abreviado.

Essa variável indica a base da árvore de documentos no servidor Web. Utilizamos o .. querendo dizer que ele é "o diretório pai" do diretório do documento raiz. Esse diretório está fora da árvore de documento, por razões de segurança. Não queremos que esse arquivo seja acessível na Web exceto pela interface que fornecemos. Esse caminho é chamado *caminho relativo* porque descreve uma posição no sistema de arquivos em relação à raiz do documento.

Como estamos atribuindo nomes abreviados às variáveis de formulário, a menos que `register_globals` esteja ativado, precisaremos da seguinte linha no início do script

```
$DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
```

para copiar o conteúdo da variável no estilo longo para o nome de estilo abreviado.

Da mesma forma como há diferentes maneiras de acessar dados de formulário, há diferentes maneiras de acessar as variáveis predefinidas de servidor. Dependendo da configuração do servidor, você pode chegar à raiz do documento por:

- `$DOCUMENT_ROOT`
- `$_SERVER['DOCUMENT_ROOT']`
- `$HTTP_SERVER_VARS['DOCUMENT_ROOT']`

Como ocorre com os dados de formulário, o primeiro estilo é o preferido.

Você também poderia especificar um caminho absoluto para o arquivo. Esse é o caminho do diretório-raiz (/ em um sistema UNIX e em geral `C:\` em um sistema Windows). Em nosso servidor UNIX, esse seria `/home/book/orders`. O problema de fazer isso é que, particularmente se você estiver hospedando seu site no servidor de outra pessoa, o caminho absoluto pode mudar. Aprendemos isso da pior maneira depois de ter de alterar caminhos absolutos em muitos scripts quando os administradores de sistema decidiram alterar a estrutura de diretórios sem aviso.

Se nenhum caminho for especificado, o arquivo será criado ou procurado no mesmo diretório que o próprio script. Isso será diferente se você estiver executando o PHP por algum tipo de empacotador CGI e dependerá da configuração do servidor.

Em um ambiente UNIX, as barras em diretórios serão barras normais (/). Se estiver utilizando uma plataforma Windows, você pode utilizar barras invertidas. Se utilizar barras invertidas, elas devem ser "escapadas" (tratadas como um carectere especial) para `fopen` entendê-las adequadamente. Para escapar um caractere, você simplesmente adiciona uma barra invertida adicional na frente dele, como é mostrado a seguir:

```
$fp = fopen("$DOCUMENT_ROOT\\..\\orders\\orders.txt", 'w');
```

Poucas pessoas utilizam barras normais nos caminhos dentro do PHP, uma vez que isso significa que o código somente funcionará no Windows. Se utilizar barras normais, freqüentemente você poderá mover o código entre máquinas Windows e UNIX sem alteração.

O segundo parâmetro de `fopen( )` é o modo de arquivo, que deve ser uma string. Isso especifica o que você quer fazer com o arquivo. Nesse caso, estamos passando `'w'` para `fopen( )` – isso significa abrir o arquivo para gravação. Um resumo dos modos de arquivo é mostrado na Tabela 2.1.

Tabela 2.1 **Resumo de modos de arquivo para `fopen( )`**

Modo	Nome do Modo Significado
r	Modo de leitura – Abre o arquivo para leitura, começando do início do arquivo.
r+	Modo de leitura – Abre o arquivo para leitura e gravação, começando do início do arquivo.
w	Modo de gravação – Abre o arquivo para gravação, começando do início do arquivo. Se o arquivo já existir, exclui o conteúdo existente. Se não existir, tenta criá-lo.
w+	Modo de gravação – Abre o arquivo para leitura e gravação, começando no início do arquivo. Se o arquivo já existir, exclui o conteúdo existente. Se não existir, tenta criá-lo.
w+	Modo de gravação cuidadosa – Abre o arquivo para gravação, começando do início do arquivo. Se o arquivo já existir, não será aberto, `fopen( )` retornará `false`, e o PHP gerará um aviso.
x	Modo de gravação cuidadosa – Abre o arquivo para gravação e leitura, começando do início do arquivo. Se o arquivo já existir, não será aberto, `fopen( )` retornará `false` , e o PHP gerará um aviso.

Tabela 2.1 Continuação

Modo	Nome do Modo Significado
a	Modo de acréscimo – Abre o arquivo somente para acrescentar (gravação), começando no final do conteúdo existente, se houver algum. Se não existir, tenta criá-lo.
a+	Modo de acréscimo – Abre o arquivo para acréscimo (gravação) e leitura, começando no final do conteúdo existente, se houver algum. Se não existir, tenta criá-lo.
b	Modo binário – Utilizado em conjunção com um dos outros modos. Talvez você queira utilizar esse se seu sistema de arquivos diferencia entre arquivos de texto e binários. Os sistemas Windows diferenciam; os sistemas UNIX não.
t	Modo de texto – Usado junto com um dos outros modos. Essa é uma opção apenas para sistemas Windows. Não é recomendada, exceto antes de você ter utilizado o código para funcionar com a opção b.

O modo de arquivo a utilizar em nosso exemplo depende de como o sistema será utilizado. Utilizamos 'w', que só permitirá que um pedido seja armazenado no arquivo. Toda vez que um novo pedido for recebido, ele sobrescreverá o pedido anterior. Provavelmente isso não é muito sensato, então é melhor especificarmos o modo de acréscimo:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
```

A função fopen( ) tem um terceiro parâmetro opcional. Você pode utilizá-lo se quiser pesquisar o include_path (definido na sua configuração de PHP – veja Apêndice A) para um arquivo. Se quiser fazer isso, configure esse parâmetro como 1. Se pedir ao PHP para pesquisar include_path, você não precisará fornecer um nome de diretório ou de caminho:

```
$fp = fopen('orders.txt', 'ab', true);
```

O quarto parâmetro também é opcional. A função fopen( ) permite que os nomes de arquivos sejam prefixados com um protocolo (como http://) e abertos em um local remoto. Alguns protocolos permitem um parâmetro extra. Veremos o uso da função fopen( ) na próxima seção deste capítulo.

Se fopen( ) abrir o arquivo com sucesso, um ponteiro é retornado para o arquivo e deve ser armazenado em uma variável, nesse caso $fp. Você utilizará essa variável para acessar o arquivo quando realmente quiser lê-lo ou gravar nele.

Abrindo arquivos por meio de FTP ou HTTP

Além de abrir arquivos locais para leitura e gravação, você pode abrir arquivos via FTP, HTTP e outros protocolos utilizando fopen( ). Mas também é possível eliminar essa possibilidade desativando a diretiva allow_url_fopen no arquivo php.ini. Se tiver problemas ao abrir arquivos remotos com fopen( ), verifique o arquivo php.ini.

Se o nome de arquivo escolhido começar com ftp://, uma conexão FTP de modo passivo será aberta com o servidor que você especificar e será retornado um ponteiro ao início do arquivo.

Se o nome de arquivo utilizado começar com http://, uma conexão HTTP será aberta com o servidor que você especificar e será retornado um ponteiro para a resposta. Ao utilizar o modo HTTP com versões mais antigas do PHP, é preciso inserir uma barra final nos nomes de diretórios, desta forma:

http://www.example.com/

e não

http://www.example.com

Quando você especificar a última forma de endereço (sem a barra), um servidor Web normalmente utilizará um HTTP a fim de redirecionar você para o primeiro endereço (com a barra). Experimente no seu navegador.

Antes do PHP 4.0.5, a função `fopen( )` não suportava redirecionamentos de HTTP; portanto, você deve especificar os URLs que fazem referência aos diretórios com uma barra final.

Com o PHP 4.3.0, você também pode abrir arquivos sobre SSL, contanto que compile ou permita suporte para OpenSSL e comece o nome do arquivo com `https://`.

Lembre-se de que os nomes de domínio no URL não fazem distinção entre letras maiúsculas e minúsculas, mas o caminho e o nome de arquivo talvez façam.

Problemas ao abrir arquivos

É possível que você cometa o erro de tentar abrir um arquivo para o qual não tem permissão de leitura ou gravação. O PHP fornecerá um aviso semelhante ao mostrado na Figura 2.2.

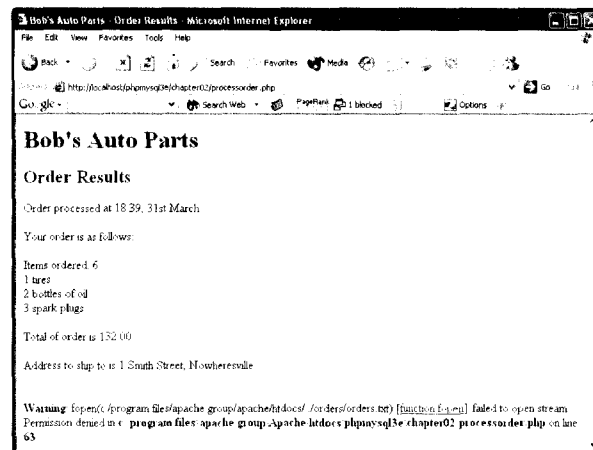



Figura 2.2 O PHP advertirá especificamente quando um arquivo não puder ser aberto.

Se obtiver esse erro, você precisa certificar-se de que o usuário sob o qual o script executa tem permissão de acessar o arquivo que está tentando utilizar. Dependendo de como o servidor estiver configurado, o script pode executar como o usuário de servidor Web ou como o proprietário do diretório em que o script está.

Na maioria dos sistemas, o script executará como o usuário de servidor Web. Se o script estivesse em um sistema UNIX no diretório `~/public_html/chapter2/`, você criaria um diretório gravável por todos no qual armazenar o pedido digitando:

```
mkdir ~/orders
chmod 777 ~/orders
```

Lembre-se de que os diretórios e arquivos nos quais qualquer pessoa grava são perigosos. Você não deve ter diretórios que são acessíveis diretamente da Web como graváveis. Por essa razão, nosso diretório `orders` está dois subdiretórios atrás, acima do diretório `public_html`. Conversaremos mais sobre segurança no Capítulo 15.

As configurações de permissão incorretas são provavelmente a causa mais comum de erro ao abrir um arquivo, mas não a única. Se o arquivo não puder ser aberto, você realmente precisa saber disso para não tentar ler os dados a partir do arquivo nem gravar dados nele.

Se a chamada para `fopen( )` falhar, a função retornará `false`. Você pode lidar com o erro de uma maneira mais amigável ao usuário suprimindo mensagem de erro do PHP e fornecendo sua própria mensagem:

```
@ $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a', 1);

  if (!$fp)
  {
    echo '<p><strong> Your order could not be processed at this time.  '
        .'Please try again later.</strong></p></body></html>';
    exit;
  }
```

O símbolo `@` na frente da chamada a `fopen( )` instrui o PHP a suprimir qualquer erro resultante da chamada de função. Normalmente é uma boa idéia saber quando os erros aconteceram, mas nesse caso vamos lidar com isso em outra parte.

Essa linha também pode ser escrita como a seguir:

```
$fp = @fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a');
```

Esse método tende a deixar menos óbvio o fato de você estar utilizando o operador de supressão de erro, portanto, pode tornar seu código mais difícil de ser depurado.

O método descrito aqui é uma maneira simplista de lidar com erros. Veremos um método mais elegante para manipulação de erros no Capítulo 7. Mas, cada coisa em seu tempo.

A instrução `if` testa a variável `$fp` para ver se um ponteiro válido de arquivo foi retornado da chamada `fopen`; se não, ela imprime uma mensagem de erro e termina a execução do script. Como a página terminará aqui, note que fechamos as tags HTML para fornecer HTML válida.

A saída ao utilizar essa abordagem é mostrada na Figura 2.3.

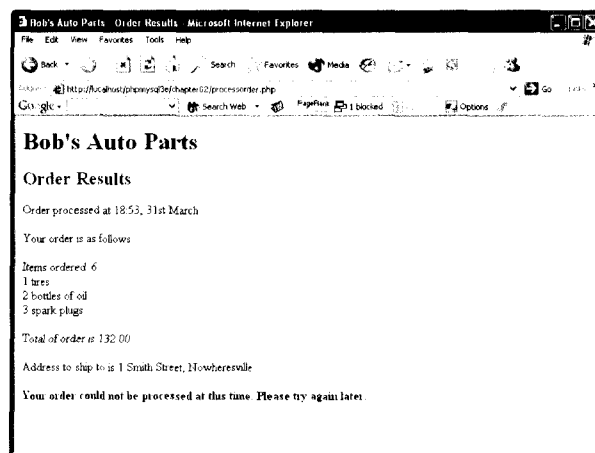



Figura 2.3 Utilizar suas próprias mensagens de erro em vez das mensagens de erro do PHP pode ser mais amigável ao usuário.

Gravando em um arquivo

Gravar dados em um arquivo no PHP é relativamente simples. Você pode utilizar as funções `fwrite( )` (file write) e `fputs( )` (file put string); `fputs( )` é um alias para `fwrite( )`. Chamamos `fwrite( )` no seguinte:

```
fwrite($fp, $outputstring);
```

Essa chamada de função diz ao PHP para gravar a string armazenada em `$outputstring` no arquivo apontado por `$fp`.

Uma nova alternativa em relação a `fwrite( )` é a função `file_put_contents( )`. Ela tem o seguinte protótipo:

```
int file_put_contents ( string nome do arquivo,
                        string dados
                        [, int flags
                        [, contexto da fonte]])
```

Essa função grava a string contida em *data* no arquivo denominado em *filename* sem qualquer necessidade de uma chamada da função fopen( ) (ou fclose( )). Ela é nova no PHP5, e faz par com file_get_contents( ), que discutiremos em breve. É mais comum utilizar os parâmetros opcionais *flags* e *contexto* ao gravar em arquivos remotos utilizando, por exemplo, HTTP ou FTP. (Veremos essas funções no Capítulo 19.)

Parâmetros para fwrite( )

A função fwrite( ) na realidade aceita três parâmetros mas o terceiro é opcional. O protótipo para fwrite( ) é:

```
int fwrite ( resource handle, string string [, int comprimento])
```

O terceiro parâmetro, *comprimento*, é o número máximo de bytes a gravar. Se esse parâmetro for fornecido, fwrite( ) gravará a *string* no arquivo apontado por *fp* até alcançar o final da *string* ou ter gravado *comprimento* bytes, o que ocorrer primeiro.

Você pode obter o tamanho da string utilizando a função embutida no PHP strlen( ), desta forma:

```
fwrite($fp, $outputstring, strlen($outputstring));
```

Você pode querer utilizar esse terceiro parâmetro ao gravar em modo binário porque ajuda a evitar alguns problemas de compatibilidade entre plataformas diferentes.

Formatos de arquivo

Quando estiver criando um arquivo de dados como o arquivo do exemplo, o formato em que você armazena os dados depende inteiramente de você. (Entretanto, se estiver planejando utilizar o arquivo de dados em outra aplicação, você pode ter de seguir as regras dessa aplicação.)

Vamos construir uma string que represente um registro em nosso arquivo de dados. Podemos fazer isso da seguinte maneira:

```
$outputstring = $date.. .$tireqty." tires \t".$oilqty." oil\t"
                .$sparkqty." spark plugs\t\$".$total
                ."\t". $address."\n";
```

Nesse exemplo simples, estamos armazenando cada registro de pedido em uma linha separada no arquivo. Escolhemos gravar um registro por linha porque isso nos permite utilizar um separador simples de registro que é o caractere de nova linha. Como as novas linhas são invisíveis, nós as representamos com a seqüência de controle "\n".

Por todo o livro, gravaremos os campos de dados no mesmo pedido toda vez e separaremos os campos com um caractere de tabulação. Novamente, como um caractere de tabulação é invisível, ele é representado pela seqüência de controle "\t". Você pode escolher qualquer delimitador que seja fácil de ler de volta.

O caractere separador ou delimitador deve ser algo que certamente não ocorrerá nas entradas; caso contrário, deveremos processar a entrada para remover ou escapar qualquer instância do delimitador. Veremos como processar a entrada no Capítulo 4. Por enquanto, assumiremos que ninguém colocará uma tabulação no nosso formulário de pedido. É difícil, mas não impossível, para o usuário colocar uma tabulação ou nova linha em um campo de entrada de HTML de única linha.

Utilizar um separador de campos especial permitirá dividir os dados de volta em variáveis separadas mais facilmente ao lermos os dados outra vez.Abordaremos isso nos Capítulos 3 e 4. Agora, trataremos cada pedido como uma única string.

Depois de processar alguns pedidos, o conteúdo do arquivo será de certo modo semelhante ao exemplo mostrado na Listagem 2.1.

Listagem 2.1 **orderx.txt – Exemplo do que o arquivo de pedidos poderia conter**

```
20:30, 31st March 4 tires  1 oil  6 spark plugs  $434.00  22 Short St, Smalltown
20:42, 31st March 1 tires  0 oil  0 spark plugs  $100.00  33 Main Rd, Newtown
20:43, 31st March 0 tires  1 oil  4 spark plugs  $26.00  127 Acacia St, Springfield
```

Fechando um arquivo

Quando terminar de utilizar um arquivo, você precisa fechá-lo. Você deve fazer isso com a função `fclose( )`da seguinte maneira:

```
fclose($fp);
```

Essa função retornará `true` se o arquivo tiver sido fechado com sucesso, ou `false` caso contrário. Geralmente, é muito menos provável que isso saia errado do que abrir um arquivo primeiro, então, nesse caso, escolhemos não testá-lo.

A listagem completa da versão final de `processorder.php` é mostrada na Listagem 2.2

Listagem 2.2 **processorder.php – Versão final do script de processamento de pedidos**

```
<?php
 // cria nomes de variáveis curtos
 $tireqty = $_POST['tireqty'];
 $oilqty = $_POST['oilqty'];
 $sparkqty = $_POST['sparkqty'];
 $address = $_POST['address'];

 $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
 <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<?php
$date = date('H:i, jS F');

echo '<p>Order processed at ';
echo $date;
echo '</p>';

echo '<p>Your order is as follows: </p>';

$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br />';

if( $totalqty == 0)
{
 echo 'You did not order anything on the previous page!<br />';
```

Listagem 2.2 Continuação

```
}
else
{
 if ( $tireqty>0 )
   echo $tireqty.' tires<br />';
 if ( $oilqty>0 )
   echo $oilqty.' bottles of oil<br />';
 if ( $sparkqty>0 )
   echo $sparkqty.' spark plugs<br />';
}

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
            + $oilqty * OILPRICE
            + $sparkqty * SPARKPRICE;

$totalamount=number_format($totalamount, 2, '.', ' ');

echo '<p>Total of order is '.$totalamount.'</p>';
echo '<p>Address to ship to is '.$address.'</p>';

$outputstring = $date."\t".$tireqty." tires \t".$oilqty." oil\t"
                .$sparkqty." spark plugs\t\$".$totalamount
                ."\t". $address."\n";

// abre o arquivo para anexar
fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');

if (!$fp)
{
 echo '<p><strong> Your order could not be processed at this time.  '
      .'Please try again later.</strong></p></body></html>';
 exit;
}

fwrite($fp, $outputstring, strlen($outputstring));
fclose($fp);

echo '<p>Order written.</p>';
?>
</body>
</html>
```

Lendo um arquivo

Neste instante, os clientes do Bob podem fazer seus pedidos via Web, mas se os funcionários do Bob quiserem ver os pedidos, terão de abrir seus próprios arquivos.

Vamos criar uma interface Web para permitir que o pessoal do Bob leia os arquivos facilmente. O código para essa interface é mostrado na Listagem 2.3.

Listagem 2.3 **vieworders.php** – A Interface de pessoal para o arquivo de pedidos

```
<?php
  //cria nome de variável abreviado
  $DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Bob's Auto Parts - Customer Orders</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Customer Orders</h2>
<?php
@  $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');

   if (!$fp)
   {
     echo '<p><strong>No orders pending.'
         .'Please try again later.</strong></p>';
     exit;
   }

   while (!feof($fp))
   {
      $order= fgets($fp, 999);
      echo $order.'<br />';
   }

   fclose($fp);
?>
</body>
</html>
```

Esse script segue a seqüência que discutimos antes: abrir, ler e fechar o arquivo. A saída desse script utilizando o arquivo de dados da Listagem 2.1 é mostrada na Figura 2.4.

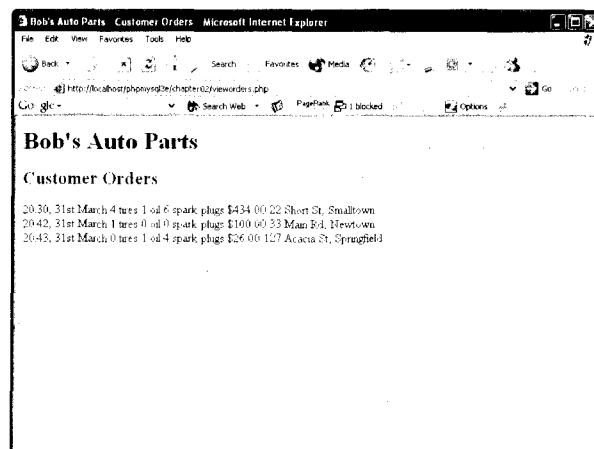



Figura 2.4 O script `vieworders.php` exibe todos os pedidos atualmente no arquivo `orders.txt` na janela de navegador.

Vamos examinar detalhadamente as funções nesse script.

Abrindo um arquivo para leitura: **fopen( )**

Novamente, abrimos o arquivo utilizando fopen( ). Nesse caso, estamos abrindo o arquivo para ler somente, então utilizamos o modo de arquivo 'rb':

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');
```

Sabendo quando parar: **feof( )**

Neste exemplo, utilizamos um loop while para ler a partir do arquivo até que o final do arquivo seja alcançado. O loop while testa o final do arquivo utilizando a função feof( ):

```
while (!feof($fp))
```

A função feof( ) aceita um ponteiro de arquivo como seu único parâmetro. Retornará true se o ponteiro de arquivo estiver no final do arquivo. Embora o nome possa parecer estranho, é fácil de lembrar se você sabe que feof significa File End Of File (Final do arquivo do arquivo).

Nesse caso (e geralmente ao ler um arquivo), lemos o arquivo até EOF ser alcançado.

Lendo uma linha por vez: **fgets( ), fgetss( )** e **fgetcsv( )**

No exemplo, utilizamos a função fgets( ) para ler o arquivo:

```
$order= fgets($fp, 999);
```

Essa função lê uma linha por vez de um arquivo. Nesse caso, ela lerá até encontrar um caractere de nova linha (\n), encontrar um EOF ou ter lido 998 bytes do arquivo. O comprimento máximo lido é o comprimento especificado menos um byte.

Há muitas funções diferentes que podem ser utilizadas para leitura de arquivos. A função fgets( ) é útil ao lidar com arquivos que contêm texto simples com o qual queremos lidar em partes.

Um variação interessante em fgets( ) é fgetss( ), que tem o seguinte protótipo:

```
string fgetss(resource fp, int comprimento, string [tags_permitidas]);
```

Isso é muito semelhante a fgets( ) exceto que vai distribuir em faixa quaisquer tags HTML e PHP localizadas na string. Se quiser deixar quaisquer tags particulares, você pode incluí-las na string *tags_permitidas*. Você utilizaria fgetss( ) por segurança ao ler um arquivo escrito por uma outra pessoa ou contendo entrada de usuário. Permitir código irrestrito de HTML no arquivo poderia mexer na sua formatação cuidadosamente planejada. Permitir o PHP irrestrito no seu servidor poderia dar rédeas à imaginação de um usuário malicioso.

A função fgetcsv( ) é outra variação de fgets( ). Ela tem o seguinte protótipo:

```
array fgetcsv ( int pa, int comprimento [, string delimitador [, string cercado_por]])
```

Ela é utilizada para dividir linhas de arquivos quando você tiver utilizado um caractere delimitador, como o caractere de tabulação que sugerimos anteriormente ou uma vírgula que é comumente utilizada por planilhas e outras aplicações. Se quisermos reconstruir as variáveis do pedido separadamente em vez de sob uma linha de texto, fgetcsv( ) permite fazer isso de modo simples. Você a chama quase da mesma maneira que chamaria fgets( ), mas passa para ela o delimitador que utilizou para separar campos. Por exemplo:

```
$order = fgetcsv($fp, 100, "\t");
```

Isso recuperaria uma linha do arquivo e a dividiria onde quer que uma tabulação (\t) fosse encontrada. Os resultados são retornados em um array ($order nesse exemplo de código). Abordaremos arrays em mais detalhe no Capítulo 3.

O parâmetro *comprimento* deve ser maior que o comprimento em caracteres da linha mais longa no arquivo que você está tentando ler.

O parâmetro *cercado_por* especifica como cada campo em uma linha é cercado. Se não especificado, o padrão é " (aspas duplas).

Lendo o arquivo inteiro: **readfile( ), fpassthru( ), file( )**

Em vez de ler uma linha por vez de um arquivo, podemos ler o arquivo inteiro de uma só vez. Há quatro maneiras diferentes para isso ser feito.

A primeira utiliza `readfile( )`. Podemos substituir o script inteiro que escrevemos anteriormente por uma linha:

```
readfile("$DOCUMENT_ROOT/../orders/orders.txt");
```

Uma chamada para a função `readfile( )` abre o arquivo, ecoa o conteúdo para saída padrão (o navegador) e então fecha o arquivo. O protótipo para `readfile( )` é:

```
int readfile(string nome_do_arquivo, int [usar_caminho_do_include])[, resource contexto]] );
```

O segundo parâmetro opcional especifica se PHP deve procurar o arquivo no *caminho_do_include* e operar da mesma maneira que em `fopen( )`. A função retorna o número total de bytes lidos do arquivo.

Em segundo lugar, você pode utilizar `fpassthru( )`. Inicialmente, você precisa abrir o arquivo que utiliza `fopen( )`. Então, pode passar o ponteiro de arquivo como argumento para `fpassthru( )`, que copiará o (isto é, fará dump do) conteúdo do arquivo a partir da posição do ponteiro para a saída padrão. A função então fecha o arquivo quando concluir.

Você pode substituir o script anterior pela `fpassthru( )` da seguinte maneira:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');
fpassthru($fp);
```

A função `fpassthru( )` retorna `true` se a leitura for bem-sucedida e `false` se não for bem-sucedida.

A terceira opção para ler o arquivo inteiro é utilizar a função `file( )`. Essa função é idêntica à `readfile( )` exceto que em vez de ecoar o arquivo para a saída padrão, ela o transforma em um array. Abordaremos isso em mais detalhe quando virmos arrays no Capítulo 3. Só como uma referência, você o chamaria utilizando

```
$filearray = file($DOCUMENT_ROOT/../orders/orders.txt");
```

Isso lerá o arquivo inteiro no array chamado `$filearray`. Cada linha do arquivo é armazenada em um elemento separado do array. Observe que essa função não é segura para binários.

Por fim, com o PHP 4.3.0, você pode utilizar a função `file_get_contents( )`. Essa função é idêntica a `readfile( )`, exceto que retorna o conteúdo do arquivo como uma string em vez de enviar a saída para o navegador. A vantagem dessa nova função é que ela é segura para binários, diferentemente da função `file( )`.

Lendo um caractere: **fgetc( )**

Outra opção para processar arquivo é ler um único caractere por vez de um arquivo. Você pode fazer isso utilizando a função `fgetc( )`. Essa função aceita um ponteiro de arquivo como seu único parâmetro e retorna o próximo caractere no arquivo. Podemos substituir o loop `while` no script original por um script que utilize `fgetc( )`:

```
while (!feof($fp))
{
  $char = fgetc($fp);
```

```
  if (!feof($fp))
    echo ($char=="\n" ? '<br />': $char);
}
```

Esse código lê um único caractere do arquivo por vez utilizando `fgetc( )` e o armazena em `$char`, até que o final do arquivo seja alcançado. Então, fazemos um pequeno processamento para substituir os caracteres de final de linha de texto, `\n`, por quebras de linha HTML, `<br />`.

Isso é simplesmente para limpar a formatação. Como os navegadores não geram uma nova linha em HTML como uma nova linha sem esse código, o arquivo inteiro seria impresso em uma única linha. (Experimente e veja.) Utilizamos o operador ternário para fazer isso elegantemente.

Um efeito colateral menor de utilizar `fgetc( )` em vez de `fgets( )` é que ele retornará o caractere `EOF` enquanto `fgets( )` não retornará. Precisamos testar `feof( )` novamente depois de lermos o caractere porque não queremos ecoar o `EOF` para o navegador.

Em geral, não é sensato ler um arquivo um caractere por vez a menos que haja alguma razão para querermos processá-lo assim.

Lendo um comprimento arbitrário: `fread( )`

A última maneira possível de ler um arquivo é utilizando a função `fread( )` para ler um número arbitrário de bytes do arquivo. Essa função tem o seguinte protótipo:

```
string fread(resource fp, int comprimento);
```

O modo como ela funciona é lendo até *`comprimento`* bytes ou até o final do arquivo, o que acontecer primeiro.

Outras funções de arquivo úteis

Há várias outras funções de arquivo que podemos utilizar que são úteis de vez em quando.

Verificando se existe ou não um arquivo: `file_exists( )`

Se quiser verificar se um arquivo existe sem realmente abri-lo, você pode utilizar `file_exists( )`, da seguinte maneira:

```
if (file_exists("$DOCUMENT_ROOT/../orders/orders.txt"))
    echo 'There are orders waiting to be processed.';
else
    echo 'There are currently no orders.';
```

Descobrindo o tamanho de um arquivo: `filesize( )`

Você pode verificar o tamanho de um arquivo com a função `filesize( )`. Ela retorna o tamanho de um arquivo em bytes:

```
echo filesize("$DOCUMENT_ROOT/../orders/orders.txt");
```

Ele pode ser utilizado em conjunção com `fread( )` para ler um arquivo inteiro (ou alguma fração do arquivo) por vez. Podemos substituir nosso script original inteiro por:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'r');
echo fread( $fp, filesize("$DOCUMENT_ROOT/../orders/orders.txt" ));
fclose( $fp );
```

A função `nl2br( )` converte os caracteres `\n` na saída para quebras de linha em HTML (`<br />`).

Excluindo um arquivo: `unlink( )`

Se quiser excluir o arquivo de pedido depois que os pedidos foram processados, você pode fazer isso utilizando `unlink( )`. (Não há nenhuma função chamada delete.) Por exemplo:

```
unlink("$DOCUMENT_ROOT/../orders/orders.txt");
```

Essa função retorna `false` se o arquivo não puder ser excluído. Isso em geral ocorrerá se as permissões no arquivo forem insuficientes ou se o arquivo não existir.

Navegando dentro de um arquivo: `rewind( )`, `fseek( )` e `ftell( )`

Você pode manipular e descobrir a posição do ponteiro de arquivo dentro de um arquivo utilizando `rewind( )`, `fseek( )` e `ftell( )`.

A função `rewind( )` redefine o ponteiro de arquivo para o começo do arquivo. A função `ftell( )` informa quanto o ponteiro avançou dentro do arquivo em bytes. Por exemplo, podemos adicionar as seguintes linhas à parte inferior do script original (antes do comando `fclose( )`):

```
echo 'Final position of the file pointer is '.(ftell($fp));
echo '<br />';
rewind($fp);
echo 'After rewind, the position is '.(ftell($fp));
echo '<br />';
```

A saída no navegador será semelhante à saída mostrada na Figura 2.5.

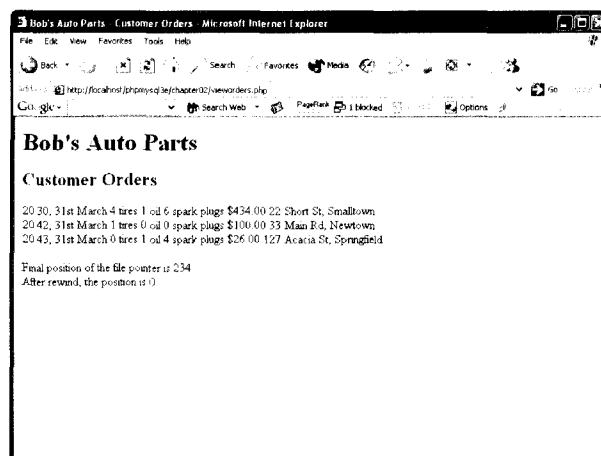



Figura 2.5 Depois de ler os pedidos, o ponteiro de arquivo aponta para o final do arquivo, um deslocamento de 234 bytes. A chamada a rewind o configura de volta para a posição 0, o início do arquivo.

A função `fseek( )` pode ser utilizada para configurar o ponteiro de arquivo para algum ponto dentro do arquivo. Seu protótipo é:

```
int fseek ( int fp, int deslocamento [, int a_partir_de])
```

Uma chamada para `fseek( )` configura o ponteiro de arquivo `fp` em um ponto iniciando de *a_partir_de* e movendo *deslocamento* bytes para o arquivo. O parâmetro *a_partir_de* opcional assume o valor `SEEK_SET` por padrão que é efetivamente o início do arquivo. Os outros possíveis valores são `SEEK_CUR` (a localização atual do ponteiro de arquivo) e `SEEK_END` (o final do arquivo).

A função `rewind( )` é equivalente a chamar a função `fseek( )` com um deslocamento zero. Por exemplo, você pode utilizar `fseek( )` para localizar um registro no meio de um arquivo ou realizar uma pesquisa binária. Freqüentemente, se você alcança o nível de complexidade em um arquivo de dados em que precisa fazer isso, sua vida será muito mais fácil se utilizar um banco de dados.

Bloqueio de arquivos

Imagine uma situação em que dois clientes estejam tentando encomendar um produto ao mesmo tempo. (Não é incomum, especialmente quando você começa a obter certo volume de tráfego em um Web site.) E se um cliente chamar `fopen( )` e começar a gravar e então o outro chamar `fopen( )` e também começar a gravar? Qual será o conteúdo final do arquivo? O conteúdo será o primeiro pedido seguido pelo segundo ou vice-versa? Será um pedido ou o outro? Ou será algo menos útil, como os dois pedidos intercalados de qualquer maneira? A resposta depende do sistema operacional, mas freqüentemente é impossível saber.

Para evitar problemas assim, você pode utilizar bloqueio de arquivo. Esse bloqueio é implementado no PHP utilizando a função `flock( )`. Essa função deve ser chamada depois que um arquivo foi aberto, mas antes de todos os dados serem lidos ou gravados no arquivo.

O protótipo para `flock( )` é:

```
bool flock (resource fp, int operação [, int &pretensobloqueio])
```

Você precisa passar para ela um ponteiro para um arquivo aberto e um número que represente o tipo de bloqueio que você requer. A função retorna `true` se o bloqueio foi adquirido com sucesso, e `false` caso contrário.

O terceiro parâmetro opcional conterá o valor `true` se o bloqueio fizer com que o processo atual seja interrompido (ou seja, se tiver de esperar).

Os possíveis valores da operação são mostrados na Tabela 2.2. Os possíveis valores mudaram no PHP 4.0.1. Os dois conjuntos de valores são mostrados na tabela.

Tabela 2.2 **Valores da operação de `flock( )`**

Valor da operação	Significado
LOCK_SH (anteriormente 1)	Bloqueio de leitura. Isso significa que o arquivo pode ser compartilhado com outros leitores.
LOCK_EX (anteriormente 2)	Bloqueio de gravação. Esse é exclusivo. O arquivo não pode ser compartilhado.
LOCK_UN (anteriormente 3)	Libera o bloqueio existente.
LOCK_NB (anteriormente 4)	A interrupção é evitada enquanto você está tentando adquirir um bloqueio.

Se for utilizar `flock( )`, você precisará adicioná-la a todos os scripts que utilizam o arquivo; caso contrário, ela é inútil.

Observe que `flock( )` não funciona com o NFS nem outros sistemas de arquivos em rede. Também não funciona com sistemas de arquivos mais antigos que não suportam bloqueamento como FAT. Em alguns sistemas operacionais, ela é implementada no nível do processo e não funcionará corretamente se você estiver utilizando uma API multiencadeada de servidor.

Para utilizá-la com esse exemplo, você pode alterar `processorder.php` da seguinte maneira:

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a');
flock($fp, LOCK_EX); // bloqueia o arquivo para gravação
fwrite($fp, $outputstring);
flock($fp, LOCK_UN); // libera bloqueio de gravação
fclose($fp);
```

Você também deve adicionar bloqueios ao `vieworders.php`:

```
$fp = fopen("$DOCUMENT_ROOT /../orders/orders.txt", 'r');
flock($fp, LOCK_SH); // bloqueia arquivo para leitura
// leitura do arquivo
```

```
flock($fp, LOCK_UN);  // libera bloqueio de leitura
fclose($fp);
```

Nosso código está agora mais poderoso, mas ainda não está perfeito. E se dois scripts tentarem adquirir um bloqueio ao mesmo tempo? Isso resultaria em uma condição de concorrência (*race condition*), onde os processos competem por bloqueios mas é incerto qual será bem-sucedido, isso poderia causar mais problemas. Podemos fazer melhor utilizando um DBMS (database management system – sistema de gerencaimento de bancos de dados).

Fazendo de uma maneira melhor: sistemas de gerenciamento de bancos de dados

Até agora todos os exemplos que vimos utilizam arquivos simples (*flat file*). Em vez disso, na próxima seção deste livro veremos como você pode utilizar MySQL, um sistema de gerenciamento de banco de dados relacional. Talvez você se pergunte: "por que devo me preocupar com isso?"

Problemas com a utilização de arquivos simples

Há vários problemas em trabalhar com arquivos simples:

- Quando um arquivo torna-se grande, seu processamento pode mostrar-se muito lento.
- É difícil procurar um registro particular ou grupo de registros em um arquivo simples. Se os registros estiverem no pedido, você pode utilizar algum tipo de pesquisa binária em conjunção com um registro de largura fixa para pesquisar em um campo-chave. Se quiser descobrir padrões nas informações (por exemplo, se você quiser localizar todos os clientes que vivem em Smalltown), você teria de ler cada registro e verificá-los individualmente.
- Lidar com acesso concorrente pode tornar-se problemático. Vimos como você pode bloquear arquivos, mas isso pode causar a condição de concorrência como discutimos antes. Além disso, pode causar um gargalo. Com bastante tráfego no site, um grupo grande de usuários pode estar esperando o arquivo ser desbloqueado para poder fazer o pedido. Se a espera for muito longa, as pessoas comprarão em outro lugar.
- Todo o processamento de arquivo que vimos até agora lida com um arquivo utilizando processamento seqüencial – isto é, começamos a partir do início do arquivo e o lemos do princípio ao fim. Se quisermos inserir ou excluir registros do meio de um arquivo (acesso aleatório), isso pode ser difícil – você termina lendo o arquivo inteiro na memória, fazendo as alterações e gravando o arquivo inteiro novamente. Com um grande arquivo de dados, isso se torna um overhead significativo.
- Além do limites oferecidos pelas permissões de arquivo, não há nenhuma maneira fácil de impor diferentes níveis de acesso a dados.

Como os RDBMSs resolvem esses problemas

Os sistemas de gerenciamento de bancos de dados relacionais resolvem todas estas questões:

- Os RDBMSs podem fornecer acesso mais rápido aos dados que aos arquivos simples. E o MySQL, o sistema de banco de dados que utilizamos neste livro, tem alguns dos benchmarks mais rápidos de qualquer RDBMS.
- Os RDBMSs podem ser facilmente consultados para extrair conjuntos de dados que satisfaçam certos critérios.

- Os RDBMSs têm mecanismos predefinidos para lidar com acesso concorrente para que você, como um programador, não tenha de se preocupar com ele.
- Os RDBMSs fornecem acessos aleatórios a seus dados.
- Os RDBMSs têm sistemas predefinidos de privilégio. O MySQL é particularmente forte nessa área.

Provavelmente, a principal razão para utilizar um RDBMS é que toda a (ou pelo menos a maior parte da) funcionalidade que você quer em um sistema de armazenamento de dados já está implementada. Com certeza, você poderia escrever sua própria biblioteca de funções de PHP, mas por que reinventar a roda?

Na Parte 2 deste livro, discutiremos como os bancos de dados relacionais geralmente funcionam, e especificamente como você pode configurar e utilizar o MySQL para criar Web sites baseados em banco de dados.

Se você está construindo um sistema simples e acha que não precisa de um banco de dados com todos os recursos, mas quer evitar bloqueio e outros problemas associados ao uso de um arquivo simples, considere usar a nova extensão SQLite do PHP. Essencialmente, ela fornece uma interface SQL a um arquivo simples. Neste livro, nosso foco é como utilizar o MySQL, mas se você quiser mais informações sobre SQLite, visite http://sqlite.org/ e http://www.php.net/sqlite.

Leitura adicional

Para obter informações adicionais sobre como interagir com o sistema de arquivos, você pode ir diretamente para o Capítulo 18. Nessa seção, examinaremos como alterar permissões, posse e nomes de arquivos; como trabalhar com diretórios e como interagir com o ambiente do sistema de arquivos.

Você também pode querer ler toda a seção de sistema de arquivos do manual on-line do PHP em http://www.php.net/filesystem.

A seguir

No próximo capítulo, discutiremos o que são arrays e como eles podem ser utilizados para processar dados nos scripts de PHP.

3

Utilizando arrays

ESTE CAPÍTULO MOSTRA COMO UTILIZAR UMA CONSTRUÇÃO de programação importante – arrays. As variáveis que vimos nos capítulos anteriores são *variáveis escalares*, que armazenam um único valor. Um *array* é uma variável que armazena um conjunto ou seqüência de valores. Um array pode ter muitos elementos. Cada elemento pode armazenar um único valor, como texto, números ou outro array. Um array contendo outros arrays é conhecido como array multidimensional.

O PHP suporta tanto arrays associativos como arrays numericamente indexados. Você provavelmente estará familiarizado com os arrays numericamente indexados se já tiver utilizado qualquer linguagem de programação, mas a menos que utilize PHP ou Perl, você pode não ter visto arrays associativos antes. Arrays associativos permitem utilizar valores mais úteis que o índice. Em vez de cada elemento ter um índice numérico, eles podem ter palavras ou outras informações significativas.

Neste capítulo, continuaremos a desenvolver o exemplo Bob's Auto Parts utilizando arrays para trabalhar mais facilmente com informações repetitivas como pedidos de cliente. Da mesma forma, escreveremos um código menor, mais compacto para fazer algumas das operações com arquivos que realizamos no capítulo anterior.

Os principais tópicos abordados neste capítulo incluem:

- Arrays indexados numericamente;
- Arrays indexados não-numericamente;
- Operadores de array;
- Arrays multidimensionais;
- Classificação de arrays;
- Funções de arrays.

O que é um array?

Vimos variáveis escalares no Capítulo 1. Uma variável escalar é uma localização identificada com um nome em que é armazenado um valor; de maneira semelhante, um array é um lugar identificado com um nome para armazenar um *conjunto* de valores, permitindo que você agrupe valores escalares.

A lista de produtos do Bob será o array para nosso exemplo. Na Figura 3.1, você pode ver uma lista de três produtos armazenados em um formato de array e uma variável, chamada `$products`, que armazena os três valores. (Veremos como criar uma variável como essa em um minuto.)

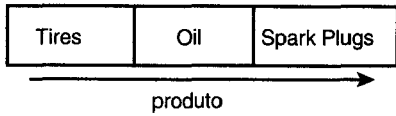



Figura 3.1 Os produtos do Bob podem ser armazenados em um array.

Depois que tivermos as informações como um array, podemos realizar várias ações úteis com o array. Utilizando as construções de loop do Capítulo 1, é possível poupar trabalho realizando as mesmas ações em cada valor no array. O conjunto inteiro das informações pode ser movido como uma unidade. Dessa maneira, com uma única linha de código, todos os valores podem ser passados para uma função. Por exemplo, poderíamos querer classificar os produtos alfabeticamente. Para tanto, poderíamos passar o array inteiro para a função `sort( )` do PHP.

Os valores armazenados em um array são chamados *elementos do array*. Cada elemento do array tem um *índice* associado (também denominado *chave*) que é utilizado para acessar o elemento. Os arrays, na maioria das linguagens de programação, têm índices numéricos que geralmente iniciam em zero ou um.

O PHP permite o uso de números e strings como índices de array. É possível usar arrays da maneira tradicional indexada numericamente ou agrupar as chaves para que a indexação seja mais significativa e útil. (Essa abordagem deve ser familiar caso você já tenha usado arrays associativos ou mapas em outras linguagens de programação.) A abordagem de programação pode variar um pouco dependendo se você está usando arrays indexados numericamente de forma padrão ou valores de índice mais interessantes.

Iniciaremos examinando arrays numericamente indexados e depois veremos as chaves definidas pelo usuário.

Arrays numericamente indexados

Esses arrays são suportados na maioria das linguagens de programação. Em PHP, os índices iniciam no zero por padrão, embora você possa alterar isso.

Inicializando arrays numericamente indexados

Para criar o array mostrado na Figura 3.1, utilize a seguinte linha de código de PHP:

```
$products = array( 'Tires', 'Oil', 'Spark Plugs' );
```

Isso criará um array chamado `$products` contendo os três valores dados – `'Tires'`, `'Oil'` e `'Spark Plugs'`. Note que, como `echo`, `array( )` é na realidade uma construção da linguagem em vez de uma função.

Dependendo do conteúdo que você precisa no array, talvez não seja necessário inicializá-lo manualmente como no exemplo anterior. Se tiver os dados de que precisa em outro array, você simplesmente pode copiar um array para outro utilizando o operador `=`.

Se quiser uma seqüência ascendente de números armazenados em um array, você pode utilizar a função `range( )` para criar automaticamente o array. A linha de código a seguir criará um array chamado `numbers` com elementos que variam de 1 a 10:

```
$numbers = range(1, 10);
```

A função `range( )` possui um terceiro parâmetro opcional que permite definir o tamanho do intervalo entre os valores. Por exemplo, se você quiser um array de números pares entre 1 e 10, pode criá-lo desta maneira:

```
$odds = range(1, 10, 2);
```

A função `range( )` também pode ser usada com caracteres, como neste exemplo:

```
$letters = range('a', 'z');
```

Se tiver as informações armazenadas em arquivo no disco, você pode carregar o conteúdo do array diretamente do arquivo. Veremos isso mais adiante neste capítulo sob o título "Carregando arrays a partir de arquivos".

Se tiver os dados para o array armazenados em um banco de dados, você poderá carregar o conteúdo de array diretamente a partir do banco de dados. Isso é abordado no Capítulo 11.

Você também pode utilizar várias funções para extrair parte de um array ou reordenar um array. Veremos algumas dessas funções mais adiante neste capítulo, sob o título "Outras manipulações de array".

Acessando o conteúdo de array

Para acessar o conteúdo de uma variável, utilize o nome dela. Se a variável for um array, acesse o conteúdo utilizando o nome da variável e uma chave ou índice. A chave ou índice indica quais valores armazenados acessamos. O índice é colocado entre colchetes depois do nome.

Digite `$products[0]`, `$products[1]` e `$products[2]` para utilizar o conteúdo do array `$products`.

Por padrão, o elemento zero é o primeiro elemento no array. Esse é o mesmo esquema de numeração utilizado em C, C++, Java e em várias outras linguagens, mas talvez você precise de algum tempo para se acostumar com ele se não conhecê-lo.

Como com outras variáveis, o conteúdo dos elementos do array é alterado utilizando o operador =. A próxima linha substituirá o primeiro elemento no array `'Tires'` por `'Fuses'`.

```
$products[0] = 'Fuses';
```

A linha a seguir poderia ser utilizada para adicionar um novo elemento – `'Fuse'` – ao final do array, fornecendo um total de quatro elementos:

```
$products[3] = 'Fuses';
```

Para exibir o conteúdo, poderíamos digitar:

```
echo "$products[0] $products[1] $products[2] $products[3]";
```

Observe que embora a análise sintática da string de PHP seja relativamente inteligente, você pode confundi-la. Se você estiver tendo problemas com arrays ou outras variáveis que não são interpretadas corretamente quando são inseridas dentro de uma string entre aspas duplas, pode colocá-las fora das aspas. A instrução `echo` anterior funcionará corretamente, mas, em muitos dos exemplos mais complexos adiante neste capítulo, você notará que as variáveis estão fora das strings entre aspas.

Como com outras variáveis de PHP, os arrays não precisam ser inicializados ou criados antecipadamente. Eles são criados de forma automática na primeira vez em que você os utiliza.

O código a seguir criará o mesmo array `$products` criado anteriormente com a instrução `array( )`:

```
$products[0] = 'Tires';
$products[1] = 'Oil';
$products[2] = 'Spark Plugs';
```

Se `$products` não existir ainda, a primeira linha criará um novo array com apenas um elemento. As linhas subseqüentes adicionam valores ao array. O array é redimensionado dinamicamente à medida que você adiciona elementos a ele. Essa funcionalidade de redimensionamento não está presente na maioria das outras linguagens de programação.

Utilizando loops para acessar o array

Como o array está indexado por uma seqüência de números, podemos utilizar um loop for mais facilmente para exibir o conteúdo:

```
for ( $i = 0; $i<3; $i++ )
  echo "$products[$i] ";
```

Esse loop fornecerá saída semelhante ao código anterior, mas exigirá menos digitação do que escrever o código manualmente para trabalhar com cada elemento em um array grande. A capacidade de utilizar um loop simples para acessar cada elemento é um excelente recurso de arrays. Também podemos usar o loop foreach, especialmente projetado para usar com os arrays. Nesse exemplo, poderíamos utilizá-lo como a seguir:

```
foreach ($products as $current)
  echo $current.' ';
```

Por sua vez, esse código armazena cada elemento na variável $current e o imprime.

Arrays com diferentes índices

No array products, permitimos que o PHP ofereça um índice padrão a cada item. Isso significa que o primeiro item que adicionamos tornou-se o item 0, o segundo item, 1 e assim por diante. O PHP também suporta arrays em que podemos associar qualquer chave ou índice que quisermos a cada valor.

Inicializando um array

O código a seguir cria um array associativo com nomes de produto como chaves, e preços como valores.

```
$prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
```

O símbolo entre as chaves e os valores é simplesmente um sinal de igual seguido imediatamente de um sinal de maior que.

Acessando os elementos do array

Novamente, acessamos o conteúdo utilizando o nome da variável e uma chave, então podemos acessar as informações que armazenamos no array de preços como $prices['Tires'], $prices['Oil'] e $prices['Spark Plugs'].

O código a seguir criará o mesmo array $prices. Em vez de criar um array com três elementos, essa versão cria um array com somente um elemento e então acrescenta outros dois.

```
$prices = array( 'Tires'=>100 );
$prices['Oil'] = 10;
$prices['Spark Plugs'] = 4;
```

Eis outra parte, ligeiramente diferente, mas equivalente de código. Nesta versão, não criamos explicitamente um array. O array é automaticamente criado quando adicionamos a ele o primeiro elemento.

```
$prices['Tires'] = 100;
$prices['Oil'] = 10;
$prices['Spark Plugs'] = 4;
```

Usando loops

Como os índices nesse array não são números, não podemos utilizar um contador simples em um loop `for` para trabalhar com o array. Podemos utilizar o loop `foreach` ou `list( )` e as construções `each( )`.

O loop `foreach` tem uma estrutura ligeiramente diferente ao utilizar arrays associativos. Podemos utilizá-lo exatamente como fizemos no exemplo anterior ou podemos incorporar as chaves também:

```
foreach ($prices as $key => $value)
  echo $key.'=>'.$value.'<br />';
```

O código a seguir lista o conteúdo de nosso array `$prices` utilizando a construção `each( )`:

```
while( $element = each( $prices ) )
{
  echo $element[ 'key' ];
  echo ' - ';
  echo $element[ 'value' ];
  echo '<br />';
}
```

A saída desse fragmento de script é mostrada na Figura 3.2.

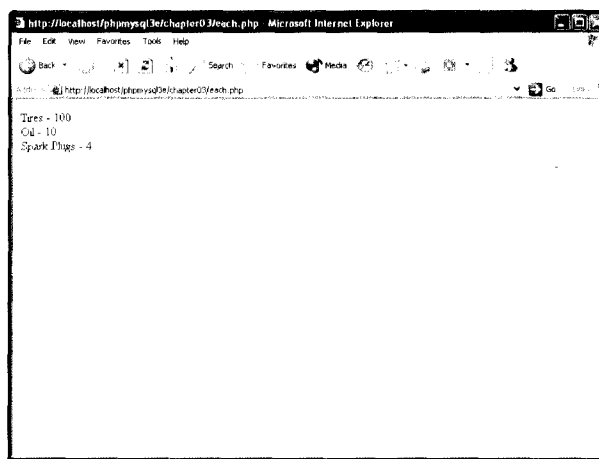



Figura 3.2 Uma instrução `each( )` pode ser utilizada para fazer um loop por arrays.

No Capítulo 1, vimos os loops `while` e a instrução `echo`. O código precedente utiliza a função `each( )`, que não utilizamos antes. Essa função retorna o elemento atual em um array e atualiza o próximo elemento. Como estamos chamando `each( )` dentro de um loop `while`, ele retorna um elemento do array de cada vez e pára quando o final do array é alcançado.

Nesse código, a variável `$element` é um array. Quando chamamos `each( )`, ela oferece um array com quatro valores e os quatro índices para as localizações de array. As localizações `key` e `0` contêm a chave do elemento atual, e as localizações `value` e `1` contêm o valor do elemento atual. Embora não faça nenhuma diferença qual delas você escolhe, escolhemos utilizar as localizações nomeadas, em vez das numeradas.

Há uma maneira mais comum e mais elegante de fazer o mesmo. A função `list( )` pode ser utilizada para dividir um array em vários valores. Podemos separar dois dos valores que a função `each( )` oferece desta maneira:

```
$list( $product, $price ) = each( $prices );
```

Essa linha utiliza each( ) para aceitar o elemento atual de $prices, retorná-lo como um array e tornar o próximo elemento o atual. Essa linha também utiliza list( ) para transformar os elementos 0 e 1 do array retornado por each( ) em duas novas variáveis chamadas $product e $price.

Podemos fazer um loop pelo array $prices inteiro, ecoando o conteúdo utilizando este script abreviado:

```
while ( list( $product, $price ) = each( $prices ) )
  echo "$product - $price<br />";
```

Esse script tem a mesma saída que o script anterior, mas é mais fácil de ler porque list( ) permite atribuir nomes às variáveis.

Algo a observar quando se utiliza each( ) é que o array monitora o elemento atual. Se quisermos utilizar o array duas vezes no mesmo script, precisamos configurar o elemento atual de volta para o início do array utilizando a função reset( ). Para fazer um loop pelo array de preços novamente, digitamos o seguinte:

```
reset($prices);
while ( list( $product, $price ) = each( $prices ) )
  echo "$product - $price<br />";
```

Esse código configura o elemento atual de volta para o início do array e permite percorrê-lo novamente.

Operadores de array

Um conjunto de operadores especiais se aplica apenas a arrays. A maioria deles possui um operador escalar análogo, como você pode ver na Tabela 3.1.

Tabela 3.1 **Operadores de array do PHP**

Operador	Nome	Exemplo	Resultado
+	União	$a + $b	União de $a e $b. O array $b é anexado a $a, mas os conflitos de chaves não são adicionados.
==	Igualdade	$a == $b	Verdadeiro se $a e $b contêm os mesmos elementos.
===	Identidade	$a === $b	Verdadeiro se $a e $b contêm os mesmos elementos na mesma ordem.
!=	Desigualdade	$a != $b	Verdadeiro se $a e $b não contêm os mesmos elementos.
<>	Desigualdade	$a <> $b	O mesmo que !=.
!==	Não-identidade	$a !== $b	Verdadeiro se $a e $b não contêm os mesmos elementos na mesma ordem.

Esses operadores são, na maior parte das vezes, bastante fáceis de serem compreendidos, mas a união requer mais explicação. O operador de união tenta adicionar os elementos de $b ao final de $a. Se os elementos em $b tiverem as mesmas chaves que alguns elementos que já estão em $a, elas não serão acrescentadas. Ou seja, nenhum elemento de $a será sobrescrito.

Você notará que todos os operadores de array da Tabela 3.1 possuem operadores equivalentes que funcionam em variáveis escalares. Contanto que você se lembre que + realiza adição em tipos escalares e união em arrays – mesmo que você não tenha interesse na aritmética por trás do comportamento – os comportamentos devem fazer sentido. Não é possível comparar arrays com tipos escalares.

Arrays multidimensionais

Os arrays não têm de ser uma lista simples de chaves e valores – cada localização no array pode armazenar outro array. Dessa maneira, podemos criar um array bidimensional. Você pode pensar em um array de duas dimensões como sendo uma matriz ou grade, com largura e altura ou linhas e colunas.

Se quiséssemos armazenar mais de uma parte de dados sobre cada produto do Bob, poderíamos utilizar um array bidimensional. A Figura 3.3 mostra produtos da loja de Bob representados como um array bidimensional com cada linha representando um produto individual e cada coluna representando um atributo do produto armazenado.

Code	Description	Price
TIR	Tires	100
OIL	Oil	10
SPK	Spark Plugs	4

produto

atributo de produto

Figura 3.3 Podemos armazenar informações adicionais sobre produtos do Bob em um array bidimensional.

Utilizando PHP, escreveríamos o código a seguir para configurar os dados no array mostrado na Figura 3.3.

```
$products = array( array( 'TIR', 'Tires', 100 ),
                   array( 'OIL', 'Oil', 10 ),
                   array( 'SPK', 'Spark Plugs', 4 ) );
```

Você pode ver a partir dessa definição que nosso array de produtos agora contém três arrays.

Para acessar os dados em um array dimensional, lembre-se de que precisamos do nome dos arrays e do índice dos elementos. Um array bidimensional é semelhante, exceto que cada elemento tem dois índices – uma linha e uma coluna. (A linha superior é a linha 0 e a coluna na extremidade esquerda é a coluna 0.)

Para exibir o conteúdo desse array, poderíamos acessar manualmente cada elemento no pedido deste modo:

```
echo '|'.$products[0][0].'|'.$products[0][1].'|'.$products[0][2].'|<br />';
echo '|'.$products[1][0].'|'.$products[1][1].'|'.$products[1][2].'|<br />';
echo '|'.$products[2][0].'|'.$products[2][1].'|'.$products[2][2].'|<br />';
```

Alternativamente, poderíamos colocar um loop `for` dentro de outro para alcançar o mesmo resultado.

```
for ( $row = 0; $row < 3; $row++ )
{
  for ( $column = 0; $column < 3; $column++ )
  {
    echo '|'.$products[$row][$column];
  }
  echo '|<br />';
}
```

As duas versões desse código produzem a mesma saída no navegador:

```
|TIR|Tires|100|
|OIL|Oil|10|
|SPK|Spark Plugs|4|
```

A única diferença entre os dois exemplos é que o código será menor se você utilizar a segunda versão com um array grande.

Você poderia preferir criar nomes de coluna em vez de números, como é mostrado na Figura 3.3. Para armazenar o mesmo conjunto de produtos, com as colunas identificadas como elas estão na Figura 3.3, você utilizaria o código a seguir:

```
$products = array( array( Code => 'TIR',
                          Description => 'Tires',
                          Price => 100
                        ),

                   array( Code => 'OIL',
                          Description => 'Oil',
                          Price => 10
                        ),
                   array( Code => 'SPK',
                          Description => 'Spark Plugs',
                          Price =>4
                        )
                 );
```

É mais fácil trabalhar com esse array se você quiser recuperar um único valor. Lembrar-se de que a descrição está armazenada na coluna Description é mais fácil do que lembrar-se de que ela está armazenada na coluna 1. Utilizando arrays associativos, você não precisa lembrar de que um item está armazenado em [x][y]. Você pode localizar facilmente os dados consultando uma localização com nomes de linha e coluna significativos.

Entretanto, perdemos a capacidade de utilizar um loop `for` simples para percorrer passo a passo cada coluna por vez. Eis uma maneira de escrever código para exibir esse array:

```
for ( $row = 0; $row < 3; $row++ )
{
  echo '|'.$products[$row]['Code'].'|'.$products[$row]['Description'].
       '|'.$products[$row]['Price'].'|<br />';
}
```

Utilizando um loop `for`, podemos percorrer o array externo numericamente indexado `$products`. Cada linha em nosso array `$products` é um array com índices descritivos. Utilizando as funções `each( )` e `list( )` em um loop `while`, podemos percorrer os arrays internos. Portanto, precisamos de um loop `while` dentro de um loop `for`.

```
for ( $row = 0; $row < 3; $row++ )
{
  while ( list( $key, $value ) = each( $products[ $row ] ) )
  {
    echo "|$value";
  }
  echo '|<br />';
}
```

Não precisamos parar nas duas dimensões – da mesma maneira que elementos do array podem armazenar novos arrays, esses novos arrays por sua vez podem armazenar mais arrays.

Um array tridimensional tem altura, largura e profundidade. Se você entende bem um array bidimensional considerando-o como uma tabela com linhas e colunas, imagine uma pilha dessas tabelas. Cada elemento será referenciado por sua camada, linha e coluna.

Se Bob dividiu os produtos em categorias, poderíamos utilizar um array tridimensional para armazená-los. A Figura 3.4 mostra produtos do Bob em um array tridimensional.

Figura 3.4 Esse array tridimensional permite dividir produtos em categorias.

A partir do código que define esse array, é possível ver que um array tridimensional é um array contendo arrays de arrays.

```
$categories = array( array ( array( 'CAR_TIR', 'Tires', 100 ),
                             array( 'CAR_OIL', 'Oil', 10 ),
                             array( 'CAR_SPK', 'Spark Plugs', 4 )
                           ),
                     array ( array( 'VAN_TIR', 'Tires', 120 ),
                             array( 'VAN_OIL', 'Oil', 12 ),
                             array( 'VAN_SPK', 'Spark Plugs', 5 )
                           ),
                     array ( array( 'TRK_TIR', 'Tires', 150 ),
                             array( 'TRK_OIL', 'Oil', 15 ),
                             array( 'TRK_SPK', 'Spark Plugs', 6 )
                           )
                   );
```

Como esse array tem apenas índices numéricos, podemos utilizar loops for aninhados para exibir seu conteúdo.

```
for ( $layer = 0; $layer < 3; $layer++ )
{
  echo "Layer $layer<br />";
  for ( $row = 0; $row < 3; $row++ )
  {
    for ( $column = 0; $column < 3; $column++ )
    {
      echo '|'.$categories[$layer][$row][$column];
    }
    echo '|<br />';
  }
}
```

Devido à maneira como os arrays multidimensionais são criados, poderíamos criar quatro, cinco ou seis dimensões de arrays. Não há nenhum limite da linguagem para o número de dimensões, mas é difícil para as pessoas visualizarem construções com mais de três dimensões. A maioria dos problemas do mundo real corresponde logicamente a construções de três ou menos dimensões.

Classificando arrays

Freqüentemente é útil classificar os dados relacionados armazenados em um array. Pegar um array dimensional e classificá-lo é bem fácil.

Utilizando sort( )

O código a seguir resulta no array sendo classificado em ordem alfabética crescente:

```
$products = array( 'Tires', 'Oil', 'Spark Plugs' );
sort($products);
```

Os elementos do array agora estarão na ordem Oil, Spark Plugs, Tires.

Também podemos classificar valores por ordem numérica. Se tivermos um array contendo os preços de produtos do Bob, podemos classificá-lo em ordem crescente numérica como mostrado a seguir:

```
$prices = array( 100, 10, 4 );
sort($prices);
```

Os preços agora estarão na ordem 4, 10, 100.

Observe que a função sort faz distinção entre letras maiúsculas e minúsculas. Todas as letras maiúsculas vêm antes de todas as letras minúsculas. Então, *A* é menor que *Z*, mas *Z* é menor que *a*.

A função também possui um segundo valor opcional. Você pode passar uma das constantes SORT_REGULAR (o padrão), SORT_NUMERIC ou SORT_STRING. A capacidade de especificar o tipo de classificação é útil quando você está comparando strings que podem conter números, por exemplo, 2 e 12. Numericamente, 2 é menor que 12, mas como string, '12' é menor que '2'.

Utilizando asort( ) e ksort( ) para classificar arrays

Se formos utilizar um array com chaves descritivas para armazenar itens e seus preços, precisamos utilizar tipos diferentes de funções de classificação para manter as chaves e os valores juntos enquanto eles são classificados.

O código a seguir cria um array contendo os três produtos e seus preços associados e então classifica o array em ordem crescente de preço.

```
$prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
asort($prices);
```

A função asort( ) ordena o array de acordo com o valor de cada elemento. No array, os valores são os preços e as chaves são as descrições textuais. Se em vez de classificar por preço quisermos classificar por descrição, utilizamos ksort( ), que classifica por chave em vez de valor. Esse código resultará nas chaves do array sendo ordenado alfabeticamente – Oil, Spark Plugs, Tires.

```
$prices = array( 'Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
ksort($prices);
```

Classificando inversamente

Você viu sort( ), asort( ) e ksort( ). Todas essas três funções de classificação diferentes classificam um array na ordem crescente. Cada uma dessas funções tem uma função de classificação inversa de correspondência para classificar um array na ordem decrescente. As versões inversas são chamadas rsort( ), arsort( ) e krsort( ).

As funções de classificação inversa são utilizadas da mesma maneira que as funções de classificação. A função rsort( ) classifica um único array numericamente indexado dimensional na ordem decrescente. A função arsort( ) classifica um array associativo dimensional na ordem decrescente utilizando o valor de cada elemento. A função krsort( ) classifica um array associativo dimensional na ordem decrescente utilizando a chave de cada elemento.

Classificando arrays multidimensionais

Classificar arrays com mais de uma dimensão ou por algo diferente da ordem alfabética ou numérica, é mais complicado. O PHP sabe comparar dois números ou duas strings de texto, mas em um array multidimensional, cada elemento é um array. O PHP não sabe comparar dois arrays, então você precisa criar um método para compará-los. Na maioria das vezes, a ordem das palavras ou números é relativamente óbvia – mas para objetos complicados, torna-se mais problemático.

Classificações definidas pelo usuário

Eis a definição de um array bidimensional que utilizamos anteriormente. Esse array armazena três produtos do Bob com um código, uma descrição e um preço para cada um.

```
$products = array( array( 'TIR', 'Tires', 100 ),
                   array( 'OIL', 'Oil', 10 ),
                   array( 'SPK', 'Spark Plugs', 4 ) );
```

Se classificarmos esse array, em que ordem os valores ficarão? Como sabemos o que o conteúdo representa, há pelo menos duas ordens úteis. Poderíamos querer classificar os produtos por ordem alfabética utilizando a descrição ou por ordem numérica do preço. Qualquer resultado é possível, mas precisamos utilizar a função `usort( )` e instruir o PHP a comparar os itens. Para fazer isso, precisamos escrever nossa própria função de comparação.

O código a seguir classifica esse array alfabeticamente utilizando a segunda coluna no array – a descrição.

```
function compare($x, $y)
{
  if ( $x[1] == $y[1] )
    return 0;
  else if ( $x[1] < $y[1] )
    return -1;
  else
    return 1;
}

usort($products, 'compare');
```

Até agora neste livro, chamamos várias das funções do PHP predefinidas. Para classificar esse array, definimos nossa própria função. Examinaremos como escrever funções em detalhes no Capítulo 5, mas aqui também fornecemos uma breve introdução.

Definimos uma função utilizando a palavra-chave `function`. Precisamos dar um nome à função. Os nomes devem ser significativos, então a chamaremos de `compare( )`. Muitas funções aceitam parâmetros ou argumentos. Nossa função `compare( )` recebe dois, um chamado `$x` e outro chamado `$y`. O propósito dessa função é considerar dois valores e determinar sua ordem.

Para esse exemplo, os parâmetros `$x` e `$y`serão dois dos arrays dentro do array principal, cada um representando um produto. Para acessar a `Description` do array `$x`, digitamos `$x[1]` porque a `Description` é o segundo elemento nesses arrays e a numeração inicia em zero. Utilizamos `$x[1]` e `$y[1]` para comparar cada `Description` dos arrays passados para a função.

Quando uma função termina, ela pode fornecer uma resposta ao código que a chamou. Isso é chamado *retornar* um valor. Para retornar um valor, utilizamos a palavra-chave `return` na função. Por exemplo, a linha `return 1;` devolve o valor 1 para o código que chamou a função.

Para ser utilizada por `usort( )`, a função `compare( )` deve comparar `$x` e `$y`. A função deve retornar 0 se `$x` for igual a `$y`, um número negativo se for menor e um número positivo se for maior. Nossa função retornará 0, 1 ou -1, dependendo dos valores de `$x` e `$y`.

A linha de código final chama a função predefinida `usort( )` com o array que devemos classificar (`$products`) e o nome da função de comparação (`compare( )`).

Se quisermos classificar o array em outra ordem, podemos simplesmente escrever uma função diferente de comparação. Para classificar por preço, precisamos ver a terceira coluna no array e criar esta função de comparação:

```
function compare($x, $y)
{
if ( $x[2] == $y[2] )
  return 0;
else if ( $x[2] < $y[2] )
  return -1;
else
  return 1;
}
```

Quando `usort($products, compare)` for chamado, o array será colocado em ordem crescente por preço.

O *u* em `usort( )` significa *usuário* porque essa função requer uma função de comparação definida pelo usuário. As versões `uasort( )` e `uksort( )` de `asort` e `ksort` também requerem uma função de comparação definida pelo usuário.

De maneira semelhante a `asort( )`, `uasort( )` deve ser utilizado ao classificar um array indexado não-numérico por valor. Utilize `asort` se seus valores são números simples ou texto. Defina uma função de comparação e utilize `uasort( )` se seus valores forem objetos mais complicados como arrays.

Semelhante a `ksort( )`, `uksort( )` deve ser utilizado ao classificar um array indexado não-numérico por chave. Utilize `ksort` se suas chaves forem números simples ou texto. Defina uma função de comparação e utilize `uksort( )` se suas chaves forem objetos mais complicados como arrays.

Classificações inversas do usuário

As funções `sort( )`, `asort( )` e `ksort( )` têm todas uma classificação inversa de correspondência com um *r* no nome de função. As classificações definidas pelo usuário não têm variantes de inversão, mas podem classificar um array multidimensional em ordem inversa. Você fornece a função de comparação, então escreve uma função de comparação que retorna os valores opostos. Para classificar pela ordem inversa, a função precisará retornar 1 se `$x` for menor que `$y`, e -1 se `$x` for maior que `$y`. Por exemplo,

```
function reverseCompare($x, $y)
{
if ( $x[2] == $y[2] )
  return 0;
else if ( $x[2] < $y[2] )
  return 1;
else
  return -1;
}
```

Chamar `usort($products, reverseCompare)` resultaria agora no array sendo colocado em ordem decrescente por preço.

Reordenando arrays

Para algumas aplicações, talvez você queira manipular a ordem do array de outras maneiras. A função `shuffle( )` reordena aleatoriamente os elementos de seu array. A função `array_reverse( )` fornece uma cópia do seu array com todos os elementos em ordem inversa.

Utilizando **shuffle( )**

Bob quer apresentar um número pequeno de seus produtos na primeira página do site. Ele tem uma grande variedade de produtos, mas gostaria que três itens aleatoriamente selecionados fossem mostrados na página inicial. Para que visitantes constantes (que retornam continuamente) não fiquem entediados, ele gostaria que os três produtos escolhidos fossem diferentes a cada visita. Ele pode facilmente realizar seu objetivo se todos os produtos estiverem em um array. A Listagem 3.1 exibe três figuras escolhidas aleatoriamente embaralhando o array em uma ordem aleatória e então exibindo os primeiros três.

Listagem 3.1 **bobs_front_page.php – Utilizando o PHP para produzir uma primeira página dinâmica para Bob's Auto Parts**

```
<?php
  $pictures = array('tire.jpg', 'oil.jpg', 'spark_plug.jpg',
                    'door.jpg', 'steering_wheel.jpg',
                    'thermostat.jpg', 'wiper_blade.jpg',
                    'gasket.jpg', 'brake_pad.jpg');

 shuffle($pictures);
?>
<html>
<head>
  <title>Bob's Auto Parts</title>
</head>
<body>
  <center>
    <h1>Bob's Auto Parts</h1>
    <table width = 100%>
      <tr>
<?php
  for ( $i = 0; $i < 3; $i++ )
  {
    echo '<td align="center"><img src="';
    echo $pictures[$i];
    echo '"width="100" height="100"></td>';
  }
?>
      </tr>
    </table>
  </center>
</body>
</html>
```

Como o código seleciona figuras aleatórias, ele produz uma página diferente toda vez que você a carrega, como mostrado na Figura 3.5.

Em versões anteriores do PHP, a função `shuffle( )` exigia que você utilizasse o gerador de números aleatórios primeiro chamando `srand( )`. Essa etapa não é mais necessária.

A função shuffle( ) não tem um histórico muito ilustre. Em versões mais antigas do PHP, ela não embaralha muito bem, fornecendo um resultado não muito aleatório. Na versão 4.2.x no Windows, ela não embaralhava nada, fornecendo um resultado que vem a ser exatamente aquilo com que você iniciou. Na versão 5, parece que funciona. Se você considera essa função importante, teste em seu servidor antes de empregá-la em suas aplicações.

Como, na verdade, você não precisa de todo o array reordenado, pode obter o mesmo resultado utilizando a função `array_rand( )`.

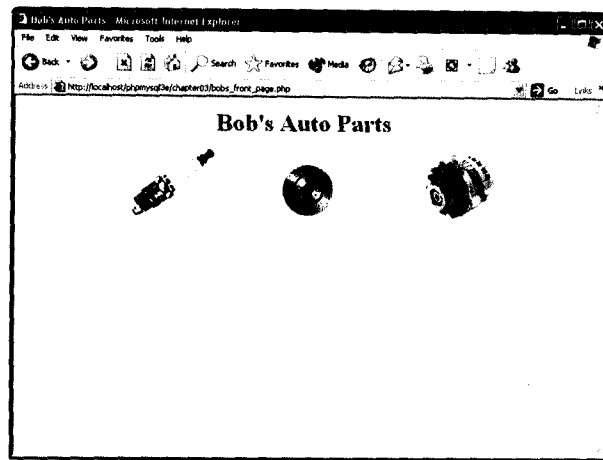



Figura 3.5 A função `shuffle( )` permite apresentar três produtos escolhidos aleatoriamente.

Utilizando array_reverse( )

A função `array_reverse( )` aceita um array e cria um novo com o mesmo conteúdo na ordem inversa. Por exemplo, há várias maneiras de criar um array contendo uma contagem regressiva de dez a um.

Utilizar `range( )` sozinho cria uma seqüência crescente, que poderíamos colocar na ordem decrescente usando `array_reverse` ou `rsort( )`. Alternativamente, poderíamos criar o array um elemento por vez escrevendo um loop `for`:

```
$numbers = array( );
for($i=10; $i>0; $i-)
  array_push( $numbers, $i );
```

Um loop `for( )` pode entrar na ordem decrescente da seguinte maneira. Configuramos um valor inicial alto e no final de cada loop utilizamos o operador – para decrementar o contador por um.

Aqui, criamos um array vazio e então utilizamos `array_push( )` para cada elemento adicionar um novo elemento ao final de um array. Como uma observação a parte, o oposto de `array_push( )` é `array_pop( )`. Essa função remove e retorna um elemento do final de um array.

Alternativamente, você pode utilizar a função `array_reverse( )` para inverter o array criado por `range( )`.

```
$numbers = range(1,10);
$numbers = array_reverse($numbers);
```

Note que `array_reverse( )` retorna uma cópia modificada do array. Como não queríamos o array original, simplesmente armazenamos a nova cópia sobre o original.

Se seus dados são apenas uma série de inteiros, você pode criá-los em ordem reversa passando –1 como o parâmetro opcional para `range`:

```
$numbers = range(10, 1, -1);
```

Carregando arrays a partir de arquivos

No Capítulo 2, armazenamos os pedidos de cliente em um arquivo. Cada linha no arquivo é algo parecido com:

```
15:42, 20th April 4 tires 1 oil 6 spark plugs $434.00 22 Short St, Smalltown
```

Para processar ou preencher esse pedido, poderíamos carregá-lo outra vez em um array. A Listagem 3.2 exibe o arquivo do pedido atual.

Listagem 3.2 **vieworders.php** – Utilizando o PHP para exibir pedidos para o Bob

```
<?php
//cria nome de variável abreviado
$DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];

$orders= file("$DOCUMENT_ROOT/../orders/orders.txt");

$number_of_orders = count($orders);
if ($number_of_orders == 0)
{
  echo '<p><strong>No orders pending.
       Please try again later.</strong></p>';
}
for ($i=0; $i<$number_of_orders; $i++)
{
  echo $orders[$i].'<br />';
}
?>
```

Esse script produz quase exatamente a mesma saída da Listagem 2.3 no capítulo anterior, que é mostrada na Figura 2.4. Dessa vez, estamos utilizando a função `file( )`, que carrega o arquivo inteiro em um array. Cada linha no arquivo torna-se um elemento de um array. Esse código também utiliza a função `count( )` para ver quantos elementos estão em um array.

Além disso, poderíamos carregar cada seção das linhas de pedidos em elementos separados do array para processar as seções separadamente ou formatá-las de modo mais atraente. A Listagem 3.3 faz exatamente isso.

Listagem 3.3 **vieworders2.php** – Utilizando o PHP para separar, formatar e exibir pedidos para o Bob

```
<?php
  //cria nome de variável abreviado
  $DOCUMENT_ROOT = $HTTP_SERVER_VARS['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Bob's Auto Parts – Customer Orders</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Customer Orders</h2>
<?php
  //Lê o arquivo inteiro.
  //Cada pedido torna-se um elemento no array
  $orders= file("$DOCUMENT_ROOT/../orders/orders.txt");
  // conta o número de pedidos no array
  $number_of_orders = count($orders);
  if ($number_of_orders == 0)
  {
    echo '<p><strong>No orders pending.
          Please try again later.</strong></p>';
  }
  echo "<table border=1>\n";
  echo '<tr><th bgcolor="#CCCCFF">Order Date</th>
          <th bgcolor="#CCCCFF">Tires</th>
          <th bgcolor="#CCCCFF">Oil</th>
          <th bgcolor="#CCCCFF">Spark Plugs</th>
          <th bgcolor="#CCCCFF">Total</th>
```

Listagem 3.3 **Continuação**

```
          <th bgcolor="#CCCCFF">Address</th>
        <tr>';
  for ($i=0; $i<$number_of_orders; $i++)
  {
    //divide cada linha
    $line = explode( . , $orders[$i] );
    // mantém somente o número de itens encomendados
    $line[1] = intval( $line[1] );
    $line[2] = intval( $line[2] );
    $line[3] = intval( $line[3] );
    // envia cada pedido para a saída
    echo "<tr><td>$line[0]</td>
              <td align="right">$line[1]</td>
              <td align="right">$line[2]</td>
              <td align="right">$line[3]</td>
              <td align="right">$line[4]</td>
              <td>$line[5]</td>
          </tr>";
  }
  echo "</table>";
?>
</body>
</html>
```

O código na Listagem 3.3 carrega o arquivo inteiro em um array, mas ao contrário do exemplo na Listagem 3.2, aqui estamos utilizando a função `explode( )` para dividir cada linha, de modo que possamos aplicar algum processamento e formatação antes da impressão. A saída desse script é mostrada na Figura 3.6.

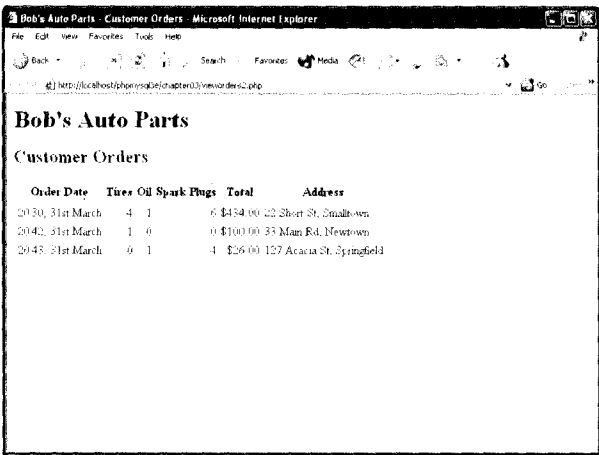



Figura 3.6 Depois de dividir os registros de pedidos com `explode`, você pode colocar cada parte de um pedido em uma célula diferente da tabela para melhorar a aparência da saída.

A função `explode` tem o seguinte protótipo:

```
array explode(string separador, string string [, int limite])
```

No capítulo anterior, utilizamos o caractere de tabulação como um delimitador ao armazenar esses dados, então aqui chamamos:

```
explode("\t" , $orders[$i] )
```

Isso “explode” em partes a string passada. Cada caractere de tabulação torna-se um divisor entre dois elementos. Por exemplo, a string

```
"15:42, 20th April\t4 tires\tl oil\t6 spark plugs\t$434.00\t
22 Short St, Smalltown"
```

é explodida nas partes `"15:42, 20th April"`, `"4 tires"`, `"1 oil"`, `"6 spark plugs"`, `"$434.00"` e `"22 Short St, Smalltown"`.

Note que o parâmetro `limit` opcional pode ser utilizado para limitar o número máximo de partes retornadas.

Não realizamos muito processamento aqui. Em vez de enviarmos para a saída o nome dos produtos em cada linha, exibimos somente o número de cada um e damos à tabela uma linha de título para mostrar o que os números representam.

Há várias maneiras como poderíamos ter extraído números dessas strings. Aqui utilizamos a função `intval( )`. Como mencionado no Capítulo 1, `intval( )` converte uma string em um inteiro. A conversão é razoavelmente inteligente e ignorará partes, como o rótulo nesse exemplo, que não possam ser convertidas em um inteiro. Abordaremos várias maneiras de processar strings no próximo capítulo.

Outras manipulações de array

Até agora, somente abrangemos cerca de metade das funções de processamento de array. Muitas outras serão úteis de vez em quando.

Navegando dentro de um array: `each( )`, `current( )`, `reset( )`, `end( )`, `next( )`, `pos( )` e `prev( )`

Mencionamos anteriormente que cada array tem um ponteiro interno que aponta para o elemento atual no array. Indiretamente utilizamos esse ponteiro antes ao utilizar a função `each( )`, mas podemos utilizar diretamente e manipular esse ponteiro.

Se criarmos um novo array, o ponteiro atual é inicializado para apontar para o primeiro elemento no array. Chamar `current( $array_name )` retorna o primeiro elemento.

Chamar `next( )` ou `each( )` avança o ponteiro em direção a um elemento. Chamar `each( $array_name )` retorna o elemento atual antes de avançar o ponteiro. A função `next( )` comporta-se de modo ligeiramente diferente – chamar `next( $array_name )` avança o ponteiro e então retorna o novo elemento atual.

Já vimos que a função `reset( )` retorna o ponteiro para o primeiro elemento no array. De maneira semelhante, chamar `end( $array_name )` envia o ponteiro para o fim do array. O primeiro e o último elementos no array são retornados por `reset( )` e `end( )`, respectivamente.

Para percorrer um array na ordem inversa, poderíamos utilizar `end( )` e `prev( )`. A função `prev( )` é a função oposta de `next( )`. Ela move o ponteiro atual um elemento para trás e então retorna o novo elemento atual.

Por exemplo, o código a seguir exibe um array em ordem inversa:

```
$value = end ($array);
while ($value)
{
  echo "$value<br />";
  $value = prev($array);
}
```

Se `$array` fosse declarado assim:

```
$array = array(1, 2, 3);
```

A saída teria a seguinte aparência em um navegador:

```
3
2
1
```

Utilizando each( ), current( ), reset( ),end( ), next( ), pos( ) e prev( ), você pode escrever seu próprio código para navegar por um array em qualquer ordem.

Aplicando qualquer função a cada elemento em um array: array_walk( )

Às vezes você pode querer trabalhar ou modificar cada elemento em um array da mesma maneira. A função array_walk( ) permite fazer isso. O protótipo de array_walk( ) é o seguinte:

```
int array_walk(array arr, string func, [mixed dados_do_usuário])
```

Semelhante à maneira como chamamos usort( ) anteriormente, array_walk( ) espera que você declare sua própria função. Como você pode ver, array_walk( ) aceita três parâmetros. O primeiro, *arr*, é o array a ser processado. O segundo, *func*, é o nome de uma função definida pelo usuário que será aplicada a cada elemento no array. O terceiro parâmetro, *dados_do_usuário*, é opcional. Se utilizá-lo, ele será passado para sua função como um parâmetro. Você verá como isso funciona em breve.

Uma função definida pelo usuário útil talvez seja uma função que exibe cada elemento com alguma formatação especificada. O próximo código exibe cada elemento em uma nova linha chamando a função definida pelo usuário myPrint( ) com cada elemento de $array:

```
function myPrint($value)
{
  echo "$value<br />";
}
array_walk($array, 'myPrint')
```

A função que você escreve precisa ter uma assinatura particular. Para cada elemento no array, array_walk obtém a chave e o valor armazenados no array e o que você tiver passado como *dados_do_usuário*, e chama sua função deste modo:

```
SuaFunção(valor, chave, dados_do_usuário)
```

Na maioria das aplicações, sua função só utilizará os valores no array. Em algumas, talvez você também precise passar um parâmetro para a função utilizando o parâmetro *dados_do_usuário*. Ocasionalmente, você pode se interessar pela chave de cada elemento bem como pelo seu valor. Sua função pode, como com MyPrint( ), escolher ignorar a chave e o parâmetro *dados_do_usuário*.

Para um exemplo ligeiramente mais complicado, escreveremos uma função que modifica os valores no array e exige um parâmetro. Observe que embora não estejamos interessados na chave, precisamos aceitá-la para receber o terceiro parâmetro.

```
function my_Multiply(&$value, $key, $factor)
{
  $value *= $factor;
}
array_walk(&$array, 'my_Multiply', 3);
```

Aqui estamos definindo uma função, my_Multiply( ), que multiplicará cada elemento no array por um fator fornecido. Precisamos utilizar o terceiro parâmetro opcional para array_walk( ) receber um parâmetro a fim de passar para nossa função e utilizá-lo como o fator pelo qual multiplicar. Como precisamos desses parâmetros, devemos definir nossa função, my_Multiply( ), para aceitar três parâmetros – o valor de um elemento do array ($value), a chave de um elemento do array ($key) e nosso parâmetro ($factor). Estamos optando por ignorar a chave.

Um ponto sutil a observar é a maneira como passamos $value. O *e* comercial (&) antes do nome variável na definição de my_Multiply( ) significa que $value *será passado por referência*. Passar por referência permite que a função altere o conteúdo do array.

Abordaremos como passar por referência em mais detalhe no Capítulo 5. Se estiver familiarizado com o termo, por enquanto simplesmente note que para passar por referência, colocamos um & antes do nome variável.

Contando elementos em um array: count( ), sizeof( ) e array_count_values( )

Utilizamos a função count( ) em um exemplo anterior para contar o número de elementos em um array de pedidos. A função sizeof( ) tem exatamente o mesmo propósito. Essas duas funções retornam o número de elementos em um array passado para elas. Você obterá uma contagem de um até o número de elementos em uma variável escalar normal, e 0 se passa um array vazio ou uma variável que não foi configurada.

A função array_count_values( ) é mais complexa. Se você chamar array_count_values($array), essa função conta quantas vezes cada valor *único* ocorre no array $array. (Isso é definido como a cardinalidade do array.) A função retorna um array associativo contendo uma tabela de freqüência. Esse array contém todos os valores únicos de $array como chaves. Cada chave tem um valor numérico que informa quantas vezes a chave correspondente ocorre no $array. Por exemplo, o código a seguir

```
$array = array(4, 5, 1, 2, 3, 1, 2, 1);
$ac = array_count_values($array);
```

cria um array chamado $ac que contém

chave	valor
4	1
5	1
1	3
2	2
3	1

Isso indica que 4, 5 e 3 ocorreram uma vez no $array, 1 ocorreu três vezes e 2 ocorreu duas vezes.

Convertendo arrays em variáveis escalares: extract( )

Se tivermos um array indexado não-numérico com vários pares de valor de chave, podemos transformá-los em um conjunto de variáveis escalares utilizando a função extract( ). O protótipo para extract( ) é da seguinte maneira:

```
extract(array array_de_var [, int tipo_de_extract] [, string prefixo] );
```

O propósito de extract( ) é aceitar um array e criar variáveis escalares com os nomes das chaves no array. Os valores dessas variáveis são configurados como os valores no array.

Eis um exemplo simples:

```
$array = array( 'key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3');
extract($array);
echo "$key1 $key2 $key3";
```

Esse código produz a seguinte saída:

```
value1 value2 value3
```

O array tinha três elementos com chaves: key1, key2 e key3. Utilizando extract( ), criamos três variáveis escalares, $key1, $key2 e $key3. A partir da saída, você pode ver que os valores de $key1, $key2 e $key3 são 'value1', 'value2' e 'value3', respectivamente. Esses valores vieram do array original.

Há dois parâmetros opcionais para extract( ): *tipo_de_extract* e *prefixo*. A variável extract_type instrui extract( ) a tratar colisões. Esses são casos em que já existe uma variável com o mesmo nome que uma chave. A resposta padrão é sobrescrever a variável existente. Quatro valores admissíveis para extract_type são mostrados na Tabela 3.2.

Tabela 3.2 **extract_types permitidos para extract( )**

Tipo	Significado
EXTR_OVERWRITE	Sobrescreve a variável existente quando ocorre uma colisão.
EXTR_SKIP	Pula um elemento quando ocorre uma colisão.
EXTR_PREFIX_SAME	Cria uma variável chamada $prefix_key quando ocorre uma colisão. Você deve fornecer prefix.
EXTR_PREFIX_ALL	Prefixa todos os nomes de variável com prefix. Você deve fornecer prefix.
EXTR_PREFIX_INVALID	Prefixa nomes variáveis que de outra maneira seriam inválidos (por exemplo, nomes variáveis numéricos) com prefix. Você deve fornecer prefix.
EXTR_IF_EXISTS	Somente extrai variáveis que já existam (isto é, que preencham variáveis existentes com valores do array). Isso foi acrescentado na versão 4.2.0 e é útil para converter, por exemplo, $_REQUEST em um conjunto de variáveis válidas.
EXTR_PREFIX_IF_EXISTS	Cria apenas uma versão prefixada se a versão não-prefixada já existir. Isso foi adicionado na versão 4.2.0.
EXTR_REFS	Extrai variáveis como referências. Isso foi adicionado na versão 4.3.0.

As duas opções mais úteis são EXTR_OVERWRITE (o padrão) e EXTR_PREFIX_ALL. As outras duas opções talvez sejam úteis ocasionalmente quando você sabe que uma colisão particular ocorrerá e deseja que a chave seja pulada ou prefixada. Um exemplo simples de como utilizar EXTR_PREFIX_ALL é mostrado a seguir. Você pode ver que as variáveis criadas são chamadas *prefixo*-sublinhado-*nomedachave*.

```
$array = array( 'key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3');
extract($array, EXTR_PREFIX_ALL, 'myPrefix');
echo "$myPrefix_key1 $myPrefix_key2 $myPrefix_key3";
```

Esse código novamente produzirá a saída: value1 value2 value3.

Note que para extract( ) extrair um elemento, a chave desse elemento deve ser um nome de variável válido, o que significa que as chaves iniciando com números ou incluindo espaços serão puladas.

Leitura adicional

Este capítulo abrange o que acreditamos ser as funções de array mais úteis do PHP. Optamos por não abranger todas as possíveis funções de array. O manual on-line do PHP disponível em http://www.php.net/array tem uma breve descrição de cada uma delas.

A seguir

No próximo capítulo, examinamos funções de processamento de string. Abordaremos funções que pesquisam, substituem, dividem e combinam strings, bem como as poderosas funções de expressão regulares que podem realizar quase todos os tipos de manipulação de string.

4

Manipulação de strings e expressões regulares

NESTE CAPÍTULO, DISCUTIREMOS COMO VOCÊ pode utilizar funções de string do PHP para formatar e manipular texto. Além disso, discutiremos como utilizar funções de string ou funções de expressões regulares para pesquisar (e substituir) palavras, frases ou outros padrões dentro de uma string.

Essas funções são úteis em muitos contextos. Com freqüência, você precisará limpar ou reformatar a entrada fornecida pelos usuários que será armazenada em um banco de dados. As funções de pesquisa são excelentes para construir aplicações de sistema de pesquisa (entre outras coisas).

Neste capítulo, abordaremos:

- Formatando strings;
- Unindo e dividindo strings;
- Comparando strings;
- Localizando e substituindo substrings com funções de string;
- Utilizando expressões regulares.

Aplicação de exemplo: Smart Form Mail

Neste capítulo, veremos funções de strings e de expressões regulares no contexto da aplicação Smart Form Mail. Adicionaremos esses scripts ao site Bob's Auto Parts que vimos nos últimos capítulos.

Desta vez, construiremos um simples e direto formulário de feedback de cliente comumente utilizado para os clientes do Bob deixarem suas queixas e elogios, como mostrado na Figura 4.1. Entretanto, nossa aplicação terá uma melhora em relação às muitas outras que você encontrará na Web. Em vez de enviar o formulário para um endereço genérico de e-mail como `feedback@example.com`, tentaremos deixar o processo mais inteligente pesquisando a entrada de frases e palavras-chave e então enviaremos o e-mail para o funcionário apropriado na empresa do Bob. Por exemplo, se o e-mail contiver a palavra "publicidade", poderíamos enviar o feedback para o departamento de Marketing. Se o e-mail vier do maior cliente do Bob, ele pode ir direto para o Bob.

Inicie com o script simples mostrado na Listagem 4.1 e o desenvolva à medida que prosseguirmos.

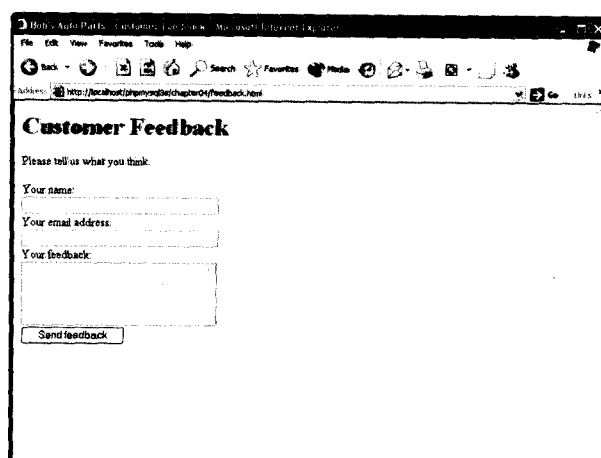



Figura 4.1 O formulário de feedback de Bob pede aos clientes seu nome, endereço de e-mail e algum comentário.

Listagem 4.1 **processfeedback.php** – Script básico para o conteúdo do formulário do e-mail

```
<?php
  //cria nome de variável abreviado
  $name=$HTTP_POST_VARS['name'];
  $email=$HTTP_POST_VARS['email'];
  $feedback=$HTTP_POST_VARS['feedback'];

  $toaddress = 'feedback@example.com';
  $subject = 'Feedback from web site';
  $mailcontent = 'Customer name: '.$name."\n"
                .'Customer email: '.$email."\n"
                ."Customer comments: \n".$feedback."\n";
  $fromaddress = 'From: webserver@example.com';

  mail($toaddress, $subject, $mailcontent, $fromaddress);
?>
<html>
<head>
  <title>Bob's Auto Parts - Feedback Submitted</title>
</head>
<body>
<h1>Feedback submitted</h1>
<p>Your feedback has been sent.</p>
</body>
</html>
```

Observe que normalmente você verificaria se os usuários preencheram todos os campos requeridos do formulário utilizando, por exemplo, `isset( )`. Omitimos isso do script e de outros exemplos para simplificar.

Neste script, você verá que concatenamos os campos de formulário e utilizamos a função `mail( )` do PHP para enviá-los por e-mail para `feedback@example.com`. Ainda não utilizamos `mail( )`; portanto, discutiremos como funciona.

Não surpreendentemente, essa função envia a mensagem de e-mail. O protótipo para `mail( )` tem a seguinte aparência:

```
bool mail(string para, string assunto, string mensagem,
        string [cabeçalhos_adicionais [, string parâmetros_adicionais]]);
```

Os três primeiros parâmetros são obrigatórios e representam o endereço para o qual enviar e-mail, a linha de assunto e o conteúdo de mensagem, respectivamente. O quarto parâmetro pode ser utilizado para enviar qualquer cabeçalho válido adicional de e-mail. Cabeçalhos válidos de e-mail são descritos no documento RFC822, que está disponível on-line se você quiser mais detalhes. (RFCs ou Request for Comment são a fonte de muitos padrões Internet – discutiremos esses padrões no Capítulo 19.) Aqui utilizamos o quarto parâmetro para adicionar um endereço "From:" à mensagem de correio. Você também pode utilizá-lo para adicionar os campos "Reply-To:" e "Cc:", entre outros. Se quiser mais de um cabeçalho adicional, simplesmente separe-os pelos caracteres de nova linha (\n\r) dentro da string, da seguinte maneira:

```
$additional_headers="From: webserver@example.com\r\n"
                    .'Reply-To: bob@example.com';
```

O quinto parâmetro opcional pode ser utilizado para passar um parâmetro para qualquer programa que você configurou para enviar mensagens de correio.

Para utilizar a função email( ), configure sua instalação de PHP para apontar para seu programa de envio de correio. Se o script não funcionar na sua forma atual, consulte o Apêndice A.

Por todo este capítulo, aprimoraremos esse script básico utilizando funções de manipulação de string e expressões regulares do PHP.

Formatando strings

Você freqüentemente precisará organizar as strings fornecidas pelos usuários (em geral a partir de uma interface de formulário HTML) antes de poder utilizá-las.

Aparando strings: chop( ), ltrim( ) e trim( )

O primeiro passo na organização é eliminar todo excesso de espaços em branco da string. Embora isso nunca seja obrigatório, pode ser útil se você vai armazenar a string em um arquivo ou banco de dados, ou se vai compará-la com outras strings.

O PHP fornece três funções úteis para esse propósito. Utilizaremos a função trim( ) para organizar nossos dados de entrada da seguinte maneira:

```
$name=trim($name);
$email=trim($email);
$feedback=trim($feedback);
```

A função trim( ) elimina os espaços em branco do início e do final de uma string e retorna a string resultante. Os caracteres que ela elimina por padrão são nova linha e retornos de carro (\n e \r), tabulações horizontais e verticais (\t e \x0B), caracteres de final de string (\0) e espaços. Você também pode passar para ele um segundo parâmetro contendo uma lista de caracteres a remover, em vez dessa lista padrão. Dependendo do seu propósito particular, você poderia gostar de utilizar em vez disso as funções ltrim( ) ou rtrim( ). As duas são semelhantes à função trim( ), no sentido de aceitar a string em questão como um parâmetro e retornar a string formatada. A diferença entre as três é que trim( ) remove os espaços em branco do início e do final de uma string, ltrim( ) remove somente espaços em branco do início (ou da esquerda) e rtrim remove espaços em branco somente do final (ou da direita).

Formatando strings para apresentação

O PHP tem um conjunto de funções que você pode utilizar para reformatar uma string de maneiras diferentes.

Utilizando formatação de HTML: a função **nl2br( )**

A função nl2br( ) recebe uma string como parâmetro e substitui todas as novas linhas pela tag XHTML
 (ou pela tag HTML <br> nas versões anteriores a 4.0.5). Isso é útil para ecoar uma string longa para o navegador. Por exemplo, utilizamos essa função para formatar o feedback do cliente e ecoá-lo:

```
<p>Your feedback (shown below) has been sent.</p>
<p><? echo nl2br($mailcontent); ?> </p>
```

Lembre-se de que a HTML ignora espaços em branco simples, então se você não filtrar essa saída por nl2br( ), ela aparecerá em uma única linha (exceto pelas novas linhas forçadas pela janela de navegador). Isso é ilustrado na Figura 4.2.

Formatando uma string para impressão

Até agora, utilizamos a construção de linguagem echo para imprimir strings para o navegador.

O PHP também suporta uma construção print( ), que faz o mesmo que echo, mas retorna um valor (true ou false, denotando sucesso).

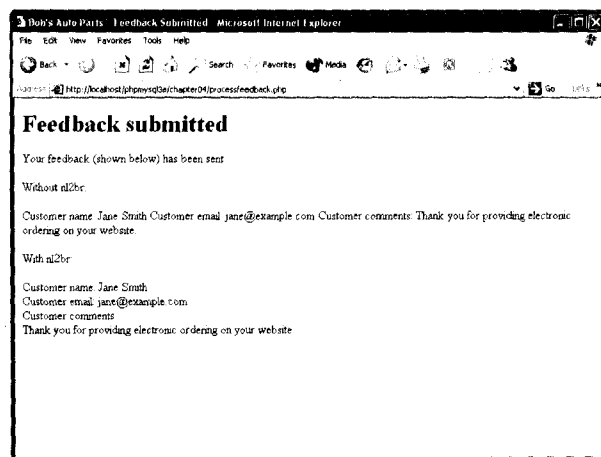



Figura 4.2 Utilizar a função nl2br( ) do PHP melhora a exibição de strings longas dentro da HTML.

Essas duas técnicas imprimem uma string *ipsis litteris*. Você pode aplicar alguma formatação mais sofisticada utilizando as funções printf( ) e sprintf( ). Essas funcionam basicamente da mesma maneira, exceto que printf( ) imprime uma string formatada para o navegador e sprintf( ) retorna uma string formatada.

Se já tiver programado em C anteriormente, você achará que essas funções são as mesmas que as das versões C. Se você não tiver, leva um certo tempo para habituar-se a elas, mas são úteis e poderosas.

Os protótipos para essas funções são:

```
string sprintf (string formato [, mixed args... ])
void printf (string formato [, mixed args...])
```

O primeiro parâmetro passado para essas duas funções é uma string de formato que descreve a forma básica da saída com códigos de formato em vez de variáveis. Os outros parâmetros são variáveis que serão substituídas pela string de formato.

Por exemplo, utilizando echo, utilizamos as variáveis que queríamos imprimir em linha, assim:

```
echo "Total amount of order is $total.";
```

Para obter o mesmo efeito com `printf( )`, você utilizaria:

```
printf ("Total amount of order is %s.", $total);
```

O %s na string de formato é chamado de especificação de conversão. Isso significa "substituir por uma string". Nesse caso, ela será substituída por `$total` interpretada como uma string. Se o valor armazenado em `$total` for 12.4, essas duas abordagens o imprimirão como `12.4`.

A vantagem de `printf( )` é que podemos utilizar uma especificação mais útil de conversão para especificar que o `$total` é realmente um número de ponto flutuante e que deve ter duas casas decimais depois do ponto de fração decimal, da seguinte maneira:

```
printf ("Total amount of order is %.2f", $total);
```

Você pode ter várias especificações de conversão na string de formato. Se tiver *n* especificações de conversão, você geralmente terá *n* argumentos depois da string de formato. Cada especificação de conversão será substituída por um argumento reformatado na ordem em que eles são listados. Por exemplo:

```
printf ("Total amount of order is %.2f (with shipping %.2f) ",
        $total, $total_shipping);
```

Aqui, a primeira especificação de conversão utilizará a variável `$total` e a segunda utilizará a variável `$total_shipping`.

Cada especificação de conversão segue o mesmo formato, que é:

```
%['caractere_de_prenchimento][-][largura][.precisão]tipo
```

Todas as especificações de conversão iniciam com um símbolo %. Se realmente quiser imprimir um símbolo %, você precisará utilizar %%.

O *caractere_de_preenchimento* é opcional. Ele será utilizado para preencher sua variável até a largura que você especificou. Um exemplo disso seria adicionar zeros iniciais a um número como um contador.

O símbolo - é opcional. Esse símbolo especifica que os dados no campo serão justificados à esquerda, em vez de alinhados à direita, o padrão.

O especificador *largura* informa a `printf( )` quanto espaço (em caracteres) deixar para a variável a ser substituída aqui.

O especificador *precisão* deve iniciar com um ponto de fração decimal. Ele deve conter o número de casas depois do ponto de fração decimal que você gostaria que fosse exibido.

A parte final da especificação é um código de tipo. Um resumo disso é mostrado na Tabela 4.1.

Tabela 4.1 **Códigos de tipo para especificação de conversão**

Tipo	Significado
b	Interpreta como um inteiro e imprime como um número binário.
c	Interpreta como um inteiro e imprime como um caractere.
d	Interpreta como um inteiro e imprime como um número decimal.
f	Interpreta como um número de dupla precisão e imprime como um número de ponto flutuante.
o	Interpreta como um inteiro e imprime como um número octal.
s	Interpreta como uma string e imprime como uma string.
x	Interpreta como um inteiro e imprime como um número hexadecimal com letras minúsculas para os dígitos a-f.
u	Interpreta como um inteiro e imprime como um decimal unsigned.
X	Interpreta como um inteiro e imprime como um número hexadecimal com letras maiúsculas para os dígitos A-F.

Na versão 4.0.6, você pode utilizar numeração de argumento, o que significa que os argumentos não precisam estar na mesma ordem das especificações de conversão. Por exemplo:

```
printf ("Total amount of order is %2\$.2f (with shipping %1\$.2f) ",
        $total_shipping, $total);
```

Simplesmente adicione a posição do argumento na lista diretamente depois do sinal %, seguido por um símbolo $ de escape – nesse exemplo 2\$ significa "substitua pelo segundo argumento na lista". Esse método também pode ser utilizado para repetir argumentos.

As duas versões alternativas dessas funções são `vprintf( )` e `vsprintf( )`. Essas variantes aceitam dois parâmetros: a string de formato e um array dos argumentos, em vez de um número variável de parâmetros.

Alterando a caixa de uma string

Você também pode reformatar a caixa (trocar maiúscula por minúscula ou vice-versa) de uma string. Isso não é particularmente útil para nossa aplicação, mas veremos alguns exemplos breves.

Se iniciarmos com a string de assunto, `$subject`, que estamos utilizando para nosso e-mail, podemos alterar sua caixa com várias funções. O efeito dessas funções é resumido na Tabela 4.2. A primeira coluna mostra o nome da função, a segunda descreve o efeito, a terceira mostra como ela seria aplicada à string `$subject` e a última coluna mostra que valor seria retornado a partir da função.

Tabela 4.2 **Funções de caixa de string e seus efeitos**

Função	Descrição	Utilização	Valor
		`$subject`	Feedback from web site
`strtoupper( )`	Coloca a string toda em letras maiúsculas	`strtoupper($subject)`	FEEDBACK FROM WEB SITE
`strtolower( )`	Coloca a string toda em letras minúsculas	`strtolower($subject)`	feedback from web site
`ucfirst( )`	Coloca o primeiro caractere de string em letra maiúscula se ele for alfabético	`ucfirst($subject)`	Feedback from web site
`ucwords( )`	Coloca o primeiro caractere de cada palavra em letra maiúscula na string que inicia com um caractere alfabético	`ucwords($subject)`	Feedback From Web Site

Formatando strings para armazenamento: `AddSlashes( )` e `StripSlashes( )`

Além de utilizarmos funções de string para reformatar uma string visualmente, podemos utilizar alguma dessas funções para reformatar strings para armazenamento em um banco de dados. Embora não abordemos realmente gravação no banco de dados até a Parte II, abordaremos agora formatação de strings para armazenamento em banco de dados.

Certos caracteres são perfeitamente válidos como parte de uma string mas podem causar problemas, particularmente ao inserir dados em um banco de dados porque este poderia interpretar esses caracteres como caracteres de controle. Os caracteres problemáticos são aspas (simples e duplas), barra invertida (\) e o caractere `NUL`.

Precisamos encontrar um caminho de marcar, ou *escapar*, esses caracteres para que bancos de dados como o MySQL possam entender o que queríamos dizer com um caractere especial literal em

vez de uma seqüência de controle. Para *escapar* esses caracteres, adicione uma barra invertida na frente deles. Por exemplo, " (aspas duplas) torna-se \" (barra invertida aspas duplas) e \ (barra invertida) torna-se \\ (duas barras invertidas). (Essa regra se aplica universalmente a caracteres especiais, então se tiver \\ em sua string, você precisará substituir por \\\\.)

O PHP fornece duas funções especificamente projetadas para caracteres de escape. Antes de gravar qualquer string em um banco de dados, você deve reformatá-las com `AddSlashes( )`, por exemplo:

```
$feedback = AddSlashes($feedback);
```

Como muitas das outras funções de string, `AddSlashes( )` aceita uma string como parâmetro e retorna a string reformatada.

A Figura 4.3 mostra os efeitos reais de utilizar essas funções na string.

Você pode experimentar essas funções em seu servidor e obter um resultado que pareça mais com a Figura 4.4.

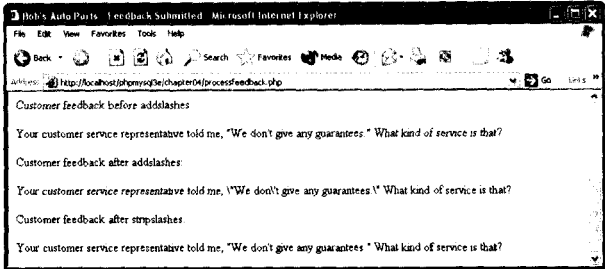



Figura 4.3 Depois de chamar a função `AddSlashes( )`, todas as aspas são precedidas por um caractere de barra invertida. `StripSlashes( )` remove as barras.

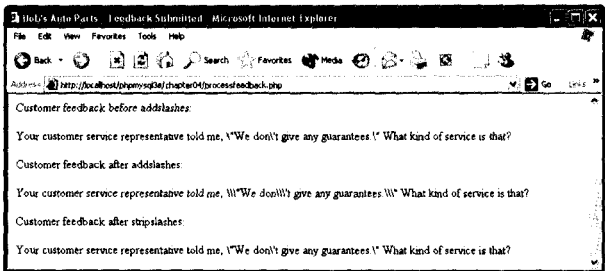



Figura 4.4 Todos os caracteres problemáticos escaparam duas vezes; Isso significa que a instrução das aspas mágicas está ativa.

Se você vir esse resultado, significa que sua configuração do PHP está definida para acrescentar e remover barras automaticamente. Essa funcionalidade é controlada pela diretiva de configuração `magic_quotes_gpc`. Por default, esse diretiva está ativada nas novas versões do PHP. As letras *gpc* são *GET*, *POST* e *cookie*. Isso significa que as variáveis vindas dessas fontes são automaticamente colocadas entre aspas. Você pode verificar se essa diretiva está ativada em seu sistema, utilizando a função `get_magic_quotes_gpc( )`, que retorna `true` se as strings dessas fontes estão sendo colocadas entre aspas automaticamente para você. Se a diretiva estiver ativada em seu sistema, é necessário chamar `stripslashes( )` antes de exibir os dados do usuário; do contrário, as barras serão exibidas.

As aspas mágicas permitem escrever código mais portável. Você pode ler mais sobre esse recurso no Capítulo 23.

Unindo e dividindo strings com funções de string

Com freqüência, queremos examinar individualmente as partes de uma string. Por exemplo, talvez queiramos examinar as palavras em uma frase (digamos para verificação ortográfica) ou dividir um

nome de domínio ou endereço de e-mail em suas partes componentes. O PHP fornece várias funções de string (e uma função de expressão regular) que permitem fazer isso.

No nosso exemplo, Bob deseja que qualquer feedback do cliente vindo de `bigcustomer.com` vá diretamente para ele, então dividiremos o endereço de e-mail que o cliente digitou em partes para descobrir se elas são do grande cliente (big customer) de Bob.

Utilizando `explode( )`, `implode( )` e `join( )`

A primeira função que poderíamos utilizar para esse propósito, `explode( )`, tem o seguinte protótipo:

```
array explode(string separador, string entrada [, int limite]);
```

Essa função aceita uma *entrada* de string e a divide em partes sobre uma string de *separador* especificada. As partes são retornadas em um array. Você pode limitar o número de fragmentos com o parâmetro *limite* opcional, adicionado ao PHP 4.0.1.

Para obter o nome de domínio do endereço de e-mail do cliente no script, podemos utilizar o seguinte código:

```
$email_array = explode('@', $email);
```

Essa chamada para `explode( )` divide o endereço de e-mail do cliente em duas partes: o nome de usuário, que é armazenado em `$email_array[0]` e o nome de domínio, que é armazenado em `$email_array[1]`. Agora podemos testar o nome de domínio para determinar a origem do cliente e então enviar seu feedback à pessoa apropriada:

```
if ($email_array[1]=='bigcustomer.com')
  $toaddress = 'bob@example.com';
else
  $toaddress = 'feedback@example.com';
```

Note que se o domínio estiver em letra maiúscula, isso não funcionará. Poderíamos evitar esse problema convertendo todo o domínio em letras maiúsculas ou em minúsculas e então verificando:

```
$email_array[1] = strtoupper ($email_array[1]);
```

Você pode inverter os efeitos de `explode( )` utilizando `implode( )` ou `join( )`, que são idênticos. Por exemplo:

```
$new_email = implode('@', $email_array);
```

Isso tira os elementos do array de `$email_array` e os une com a string passada no primeiro parâmetro. A chamada de função é muito semelhante a `explode( )`, mas o efeito é oposto.

Utilizando `strtok( )`

Diferente de `explode( )`, que divide uma string em todas suas partes de uma vez, `strtok( )` obtém partes (chamadas *tokens*) de uma string uma por vez. `strtok( )` é uma alternativa útil para utilizar `explode( )` para processar palavras de uma string por vez.

O protótipo para `strtok( )` é:

```
string strtok(string entrada, string separador);
```

O separador pode ser um caractere ou uma string de caracteres, mas observe que a string de entrada será dividida em cada um dos caracteres na string separador em vez de na string separador inteira (como `explode` faz).

Chamar `strtok( )` não é tão simples quanto parece no protótipo. Para obter o primeiro token de uma string, você chama `strtok( )` com a string que deseja usar como token e um separador. Para obter os tokens subseqüentes da string, você simplesmente passa um único parâmetro – o separa-

dor. A função mantém seu próprio ponteiro interno em seu lugar na string. Se quiser redefinir o ponteiro, você pode passar a string para ele novamente.

A função strtok( ) é em geral utilizada da seguinte maneira:

```
$token = strtok($feedback, ' ');
echo $token.'<br />';
while ($token!='')
{
  $token = strtok(' ');
  echo $token.'<br />';
};
```

Como de costume, é uma boa idéia verificar se o cliente realmente digitou algum feedback no formulário, utilizando, por exemplo, empty( ). Omitimos essas verificações para sermos mais breves.

Isso imprime cada token do feedback do cliente em uma linha separada e faz loop até que não haja mais tokens. Observe que antes da versão 4.1.0 do PHP, a função strtok( ) do PHP não funcionava exatamente da mesma maneira que a função correspondente no C. Se houver duas instâncias de um separador *em seqüência* na sua string alvo (nesse exemplo, dois espaços em seqüência), strtok( ) retorna uma string vazia. Não é possível diferenciar isso a partir da string vazia retornada quando você chegar ao final da string alvo. Além disso, se um dos tokens for 0, uma string vazia será retornada. Isso tornou strtok( ) do PHP um pouco menos útil que aquele em C. A nova versão funciona corretamente, pulando strings vazias.

Utilizando substr( )

A função substr( ) permite acessar uma substring entre determinados pontos iniciais e finais de uma string. Essa função não é apropriada para nosso exemplo, mas pode ser útil quando você precisar obter partes de string de formato fixo.

A função substr( ) tem o seguinte protótipo:

```
string substr(string string, int início[, int comprimento] );
```

Essa função retorna uma substring copiada de dentro da *string*.

Veremos exemplos utilizando esta string de teste:

```
$test = 'Your customer service is excellent';
```

Se chamá-la com um número positivo para *início* (somente), você obterá a string a partir da posição *início* até o final da string. Por exemplo,

```
substr($test, 1);
```

retorna our customer service is excellent. Observe que a posição de string inicia em 0, como acontece com arrays.

Se chamar substr( ) com um *iniciar* negativo (somente), você obterá a string do final da string menos *iniciar* caracteres do final da string. Por exemplo,

```
substr($test, -9);
```

retorna excellent

O parâmetro de comprimento pode ser utilizado para especificar um número de caracteres a retornar (se ele for positivo) ou o caractere final da seqüência de retorno (se ele for negativo). Por exemplo,

```
substr($test, 0, 4);
```

retorna os primeiros quatro caracteres da string, a saber, Your. O código a seguir,

```
echo substr($test, 4, -13);
```

retorna os caracteres entre o quarto caractere, da esquerda para a direita, e o décimo terceiro, da direita para a esquerda, isto é, `customer service`. O primeiro caractere é a posição 0, então a posição 5 é o sexto caractere.

Comparando strings

Até agora utilizamos == para comparar a igualdade entre duas strings. Podemos fazer algumas comparações ligeiramente mais sofisticadas utilizando o PHP. Dividimos essas comparações em duas categorias: correspondências parciais e outros. Lidaremos com outros primeiro e então nos concentraremos na correspondência parcial, que será necessária para desenvolver mais ainda o exemplo Smart Form.

Ordenação de string: **strcmp( )**, **strcasecmp( )** e **trnatcmp( )**

Essas funções podem ser utilizadas para ordenar strings. Isso é útil ao classificar dados.

O protótipo para `strcmp( )` é:

```
int strcmp(string str1, string str2);
```

A função espera receber duas strings, que ela comparará. Se elas forem iguais, retornará 0. Se `str1` vier depois (ou for maior) que `str2` na ordenação lexicográfica, `strcmp( )` retornará um número maior que zero. Se `str1` for menor que `str2`, `strcmp( )` retornará um número menor que zero. Essa função diferencia letras maiúsculas de minúsculas.

A função `strcasecmp( )` é idêntica exceto que não faz distinção entre letras maiúsculas e minúsculas.

A função `strnatcmp( )` e sua gêmea que não faz distinção entre letras maiúsculas e minúsculas, `strnatcasecmp( )`, foram adicionados no PHP 4. Essas funções comparam strings de acordo com uma "ordenação natural", que é mais parecida com a maneira como um humano compararia. Por exemplo, `strcmp( )` ordenaria a string `"2"` maior que a string `"12"` porque a primeira é lexicograficamente maior. `strnatcmp( )` faria isso de maneira oposta. Você pode ler mais sobre ordenação natural em:

http://www.naturalordersort.org/

Testando comprimento da string com **strlen( )**

Podemos verificar o comprimento de uma string com a função `strlen( )`. Se você passar uma string para essa função, ela retornará seu comprimento. Por exemplo, `strlen('hello')` retorna 5.

Isso pode ser utilizado para validar dados de entrada. Considere o endereço de e-mail em nosso formulário, armazenado em `$email`. Uma maneira básica de validar um endereço de e-mail armazenado em `$email` é verificando seu comprimento. No meu entender, o comprimento mínimo de um endereço de e-mail é seis caracteres – por exemplo, *a@a.to* se você tiver um código do país sem domínios de segundo nível, nome de servidor de uma letra e endereço de e-mail de uma letra. Portanto, um erro poderia ser produzido se o endereço não tivesse pelo menos esse comprimento:

```
if (strlen($email) < 6)
{
  echo 'That email address is not valid';
  exit;  // termina a execução do script de PHP
}
```

Claramente, essa é uma maneira muito simplista de validar essas informações. Veremos maneiras melhores na próxima seção.

Localizando e substituindo substrings com funções de string

É comum querer verificar se uma substring particular está presente em uma string maior. Essa correspondência parcial normalmente é mais útil que testar quanto à igualdade.

No nosso exemplo Smart Form, queremos procurar certas frases-chave no feedback do cliente e enviar uma mensagem de correio para o departamento apropriado. Se quisermos enviar mensagens de e-mail a respeito das lojas do Bob para o gerente das lojas de varejo, queremos saber se a palavra "shop" (loja) (ou uma palavra derivada) aparece na mensagem.

Dadas as funções que já vimos, poderíamos utilizar `explode( )` ou `strtok( )` para recuperar as palavras individuais na mensagem e então compará-las utilizando o operador `==` ou `strcmp( )`.

Mas também poderíamos fazer a mesma coisa com uma única chamada de função para uma das funções de localização de string ou expressão regular. Essas funções são utilizadas para procurar um padrão dentro de uma string. Veremos um por um cada conjunto de funções.

Localizando strings em strings: strstr( ), strchr( ), strrchr( ), stristr( )

Para localizar uma string dentro de outra string você pode utilizar qualquer das funções `strstr( )`, `strchr( )`, `strrchr( )` ou `stristr( )`.

A função `strstr( )` é a mais genérica e pode ser utilizada para localizar uma correspondência de string ou caractere dentro de uma string mais longa. Observe que no PHP, a função `strchr( )` é exatamente a mesma que `strstr( )`, embora seu nome implique que ela seja utilizada para localizar um caractere em uma string, semelhante à versão em C dessa função. No PHP, qualquer uma dessas funções pode ser utilizada para localizar uma string dentro de uma string, incluindo localizar uma string contendo somente um único caractere.

O protótipo para `strstr( )` é:

```
string strstr(string palheiro, string agulha);
```

Você passa para a função um *palheiro* a ser pesquisado e uma *agulha* a ser localizada. Se uma correspondência exata da *agulha* for encontrada, a função retorna o *palheiro* de *agulha* para a frente; caso contrário, retorna `false`. Se a *agulha* ocorrer mais de uma vez, a string retornada iniciará a partir da primeira ocorrência de *agulha*.

Por exemplo, no aplicação Smart Form, podemos decidir para onde enviar o e-mail da seguinte maneira:

```
$toaddress = 'feedback@example.com';  // o valor padrão

// Altera $toaddress se os critérios forem atendidos
if (strstr($feedback, 'shop'))
  $toaddress = 'retail@example.com';
else if (strstr($feedback, 'delivery'))
  $toaddress = 'fulfilment@example.com';
else if (strstr($feedback, 'bill'))
    $toaddress = 'accounts@example.com';
```

Esse código verifica certas palavras-chave no feedback e envia a mensagem de correio à pessoa apropriada. Se, por exemplo, o feedback do cliente for "I still haven't received delivery of my last order" ("Ainda não recebi meu último pedido"), a string "delivery" será detectada e o feedback será enviado para `fulfilment@example.com`.

Há duas variantes em `strstr( )`. A primeira variante é `stristr( )`, que é quase idêntica, mas não faz distinção entre letras maiúsculas e minúsculas. Isso será útil para essa aplicação uma vez que o cliente poderia digitar `'delivery'`, `'Delivery'` ou `'DELIVERY'`.

A segunda variante é `strrchr( )`, que, novamente, é quase idêntica, mas essa variante retornará o *palheiro* a partir da última ocorrência de *agulha* para a frente.

Localizando a posição de uma substring: strpos( ), strrpos( )

As funções strpos( ) e strrpos( ) operam de um modo semelhante a strstr( ), exceto que em vez de retornar uma substring, elas retornam a posição numérica de uma *agulha* dentro de um *palheiro*.

A função strpos( ) tem o seguinte protótipo:

```
int strpos(string palheiro, string agulha, int [deslocamento] );
```

O inteiro retornado representa a posição da *primeira* ocorrência da *agulha* dentro do *palheiro*. O primeiro caractere está na posição 0 como de costume.

Por exemplo, o código a seguir ecoará o valor 4 para o navegador:

```
$test = 'Hello world';
echo strpos($test, 'o');
```

Nesse caso, passamos somente um único caractere como *agulha*, mas esse caractere pode ser uma string de qualquer comprimento.

O parâmetro opcional de *deslocamento* é utilizado para especificar um ponto dentro do *palheiro* para iniciar a pesquisa. Por exemplo:

```
echo strpos($test, 'o', 5);
```

Esse código ecoará o valor 7 para o navegador porque o PHP começou a procurar o caractere o na posição 5 e, portanto, não vê o caractere na posição 4.

A função strrpos( ) é quase idêntica, mas retornará a posição da última ocorrência de *agulha* no *palheiro*. Diferente de strpos( ), ela só funciona com uma *agulha* de um único caractere. Portanto, se passar para ela uma string como *agulha*, ela só utilizará o primeiro caractere da string para correspondência.

Em qualquer desses casos, se *agulha* não estiver na string, tanto strpos( ) como strrpos( ) retornarão false. Isso pode ser problemático porque false em uma linguagem de tipificação fraca como o PHP é equivalente a 0, isto é, o primeiro caractere em uma string.

Você pode evitar esse problema utilizando o operador === para testar valores de retorno:

```
$result = strpos($test, 'H');
if ($result === false)
  echo 'Not found'
else
  echo 'Found at position $result';
```

Observe que isso só funcionará no PHP 4 e superior – em versões anteriores você pode testar false testando o valor de retorno para ver se ele é uma string (isto é, false).

Substituindo substrings: str_replace( ), substr_replace( )

A funcionalidade de localizar e substituir pode ser extremamente útil para strings. No passado, utilizamos localizar e substituir para personalizar documentos gerados pelo PHP – por exemplo substituindo <nome> pelo nome de uma pessoa e *<endereço>* por seu endereço. Você também pode utilizar isso para censurar termos particulares, como em uma aplicação de fórum de discussão ou mesmo na aplicação Smart Form. Novamente, você pode utilizar funções de string ou funções de expressões regulares para esse propósito.

A função de string mais comumente utilizada para substituição é str_replace( ). Ela tem o seguinte protótipo:

```
mixed str_replace(mixed agulha, mixed nova_agulha, mixed palheiro);
```

Essa função substituirá todas as instâncias de *agulha* em *palheiro* por *nova_agulha* e retornará a nova versão do *palheiro*.

Nota Com o PHP 4.0.5 você pode passar todos os parâmetros como arrays e a função funcionará de maneira notavelmente inteligente. Você pode passar um array de palavras a serem substituídas, um array de palavras substitutas (respectivamente) e um array de strings em que aplicar essas regras. A função então retornará um array de strings revisadas.

Por exemplo, como as pessoas podem utilizar o Smart Form para fazer queixas, talvez utilizem algumas palavras de baixo calão. Como programadores, podemos proteger os vários departamentos do Bob dessa maneira:

```
$feedback = str_replace($offcolor, '%!@*', $feedback);
```

A função `substr_replace( )` é utilizada para localizar e substituir uma substring particular de uma string baseada na sua posição. Ela tem o seguinte protótipo:

```
string substr_replace(string string, string substituta, int início, int [comprimento] );
```

Essa função substituirá parte da string *string* pela string *substituta*. Qual parte é substituída depende dos valores dos parâmetros *início* e *comprimento* opcionais.

O valor `start` representa um deslocamento na string onde a substituição deve iniciar. Se ele for 0 ou positivo, trata-se de um deslocamento desde o início da string; se for negativo, trata-se de um deslocamento a partir do final da string. Por exemplo, essa linha de código substituirá o último caractere em `$test` por `"X"`:

```
$test = substr_replace($test, 'X', -1);
```

O valor *comprimento* é opcional e representa o ponto em que o PHP parará a substituição. Se você não fornecer esse valor, a string será substituída do *start* para o final da string.

Se *comprimento* for zero, a string de substituição realmente será *inserida* na string sem sobrescrever a string existente. Um *comprimento* positivo representa o número de caracteres que você deseja que sejam substituídos pela nova string.

Um *comprimento* negativo representa o ponto em que você gostaria de parar de substituir os caracteres, contados a partir do final da string.

Introdução às expressões regulares

O PHP suporta dois estilos de sintaxe de expressão regular: POSIX e Perl. O estilo POSIX da expressão regular é compilado no PHP por padrão, mas você pode utilizar o estilo Perl compilando na biblioteca PCRE ("Perl-compatible regular expression", expressão regular compatível com Perl). Abordaremos o estilo POSIX mais simples, mas se você já for programador de Perl ou quiser aprender mais sobre PCRE, leia o manual on-line em http://php.net.

Nota As expressões regulares do POSIX são mais fáceis de aprender e são executadas mais rapidamente, mas não são seguras para binários.

Até agora, toda a correspondência de padrão que fizemos utilizou as funções de string. Estávamos limitados à correspondência exata ou correspondência exata de substring. Se quiser fazer correspondência de padrão mais complexa, você deve utilizar expressões regulares. As expressões regulares são difíceis de compreender a princípio, mas podem ser extremamente úteis.

Os princípios básicos

Uma expressão regular é uma maneira de descrever um padrão em uma parte de texto. As correspondências exatas (ou literais) que fizemos até agora são uma forma de expressão regular. Por exemplo, anteriormente estávamos procurando termos de expressão regular como `"shop"` e `"delivery"`.

Localizar expressões regulares no PHP é mais parecido com uma correspondência de `strstr( )` do que com uma comparação de igualdade porque você está procurando uma string em algum lugar dentro de outra string. (Pode ser em qualquer lugar dentro dessa string a menos que você especifique o contrário.) Por exemplo, a string "shop" localiza a expressão regular "shop". Ela também localiza as expressões regulares "h", "ho" e assim por diante.

Podemos utilizar caracteres especiais para indicar um meta-significado além de localizar caracteres exatamente. Por exemplo, com caracteres especiais você pode indicar que um padrão deve ocorrer no início ou no final de uma string, qual parte de um padrão pode ser repetida ou quais caracteres em um padrão devem ser de um tipo particular. Você também pode localizar ocorrências literais de caracteres especiais. Examinaremos cada um desses.

Conjuntos e classes de caracteres

Utilizar os conjuntos de caracteres fornece imediatamente expressões regulares mais poderosas que expressões de correspondência exata. Os conjuntos de caracteres podem ser utilizados para localizar qualquer caractere de um *tipo* particular – eles são realmente um tipo de curinga.

Antes de tudo, você pode utilizar o caractere . como um curinga para qualquer outro caractere único exceto um caractere de nova linha (\n). Por exemplo, a expressão regular

```
.at
```

identifica as strings 'cat', 'sat' e 'mat', entre outras. Esse tipo de correspondência de curinga é freqüentemente utilizado para correspondência de nome de arquivo nos sistemas operacionais.

Entretanto, com expressões regulares, você pode ser mais específico sobre o tipo de caractere que gostaria de localizar e pode realmente especificar um conjunto a que um caractere deve pertencer. No exemplo anterior, a expressão regular identifica 'cat' e 'mat', mas também identifica '#at'. Se quiser limitar isso para um caractere entre a e z, você pode especificá-lo da seguinte maneira:

```
[a-z]at
```

Qualquer coisa incluída entre *colchetes especiais* [e] é uma classe de caractere – um conjunto de caracteres ao qual um caractere procurado deve pertencer. Observe que a expressão nos colchetes localiza somente um único caractere.

Você pode listar um conjunto; por exemplo:

```
[aeiou]
```

significa qualquer vogal.

Você também pode descrever um intervalo, como fazemos aqui utilizando o caractere especial hífen, ou um conjunto de intervalos:

```
[a-zA-Z]
```

Esse conjunto de intervalos significa qualquer caractere alfabético em letra maiúscula ou minúscula.

Você também pode utilizar conjuntos para especificar que um caractere não pode ser membro de um conjunto. Por exemplo,

```
[^a-z]
```

localiza qualquer caractere que *não* esteja entre a e z. O símbolo circunflexo significa *não* quando está colocado entre colchetes. Esse símbolo tem outro significado quando utilizado fora de colchetes, que veremos em um minuto.

Além de listar conjuntos e intervalos, várias *classes de caractere* predefinidas podem ser utilizadas em uma expressão regular. Essas classes são mostradas na Tabela 4.3.

Tabela 4.3 **Classes de caractere para utilização nas expressões regulares no estilo POSIX**

Classe	Localiza
`[[:alnum:]]`	Caracteres alfanuméricos
`[[:alpha:]]`	Caracteres alfabéticos
`[[:lower:]]`	Letras minúsculas
`[[:upper:]]`	Letras maiúsculas
`[[:digit:]]`	Dígitos decimais
`[[:xdigit:]]`	Dígitos hexadecimais
`[[:punct:]]`	Pontuação
`[[:blank:]]`	Guias e espaços
`[[:space:]]`	Caracteres de espaços em branco
`[[:cntrl:]]`	Caracteres de controle
`[[:print:]]`	Todos os caracteres imprimíveis
`[[:graph:]]`	Todos os caracteres imprimíveis exceto espaço

Repetição

Freqüentemente você quer especificar que existem diversas ocorrências de uma string particular ou classe de caractere. Você pode representar isso utilizando dois caracteres especiais na sua expressão regular. O símbolo * significa que o padrão pode ser repetido zero ou mais vezes e o símbolo + significa que o padrão pode ser repetido uma ou mais vezes. O símbolo deve aparecer logo depois da parte da expressão em que ele é aplicado. Por exemplo:

```
[[:alnum:]]+
```

significa "pelo menos um caractere alfanumérico".

Subexpressões

Muitas vezes é útil ser capaz de dividir uma expressão em subexpressões de modo que você possa, por exemplo, representar "pelo menos uma destas strings seguidas exatamente por uma daquelas". Você pode fazer isso utilizando parênteses, exatamente da mesma maneira que utilizaria em uma expressão aritmética. Por exemplo,

```
(very )*large
```

identifica `'large'`, `'very large'`, `'very very large'` e assim por diante.

Subexpressões contadas

Podemos especificar quantas vezes algo pode ser repetido utilizando uma expressão numérica entre chaves ({ }). Você pode mostrar um número exato de repetições (`{3}` significa exatamente três repetições), um intervalo de repetições (`{2, 4}` significa de 2 a 4 repetições) ou um intervalo aberto sem restrições de repetições (`{2,}` significa pelo menos duas repetições).

Por exemplo,

```
(very ){1, 3}
```

localiza `'very '`, `'very very '` e `'very very very '`.

02/09/2008
03/09/08

Ancorando ao início ou ao final de uma string

O padrão [a-z] localizará qualquer string contendo um caractere minúsculo alfabético. Não interessa se a string é um caractere longo ou contém um único caractere localizado em uma string mais longa.

Você também pode especificar se uma subexpressão particular deve aparecer no início, no final ou em ambos. Isso é muito útil quando você quer certificar-se de que somente seu termo de pesquisa e nada mais apareça na string.

O símbolo circunflexo (^) é utilizado no início de uma expressão regular para mostrar que ele deve aparecer no começo de uma string pesquisada, e $ é utilizado no final de uma expressão regular para mostrar que ele deve aparecer no final.

Por exemplo, a expressão a seguir identifica bob no início de uma string:

```
^bob
```

A expressão a seguir localiza com no final de uma string:

```
com$
```

Por fim, isso localiza qualquer caractere individual de *a* a *z* na string:

```
^[a-z]$
```

Ramificação

Você pode representar uma escolha em uma expressão regular com um pipe vertical. Por exemplo, se quiser identificar com, edu ou net, você pode utilizar a expressão:

```
(com)|(edu)|(net)
```

Localizando caracteres especiais literais

Se quiser localizar um dos caracteres especiais mencionados nesta seção, como ., {, ou $, você deve colocar uma barra invertida (\) na frente dele. Se quiser representar uma barra, você deve usar duas barras invertidas (\\).

Tenha cuidado ao colocar seus padrões de expressões regulares em strings com aspas simples no PHP. Usar expressões regulares em strings com aspas duplas no PHP pode gerar complicações desnescessárias. O PHP também utiliza a barra invertida para escapar caracteres especiais – como a barra invertida. Se quiser inserir uma barra invertida em seu padrão, precisa usar duas para indicar que é uma barra invertida literal, e não um código de escape.

Da mesma forma, se quiser uma barra invertida literal em uma string com aspas duplas no PHP, precisa utilizar duas, pela mesma razão. O resultado cumulativo e um pouco confuso dessas regras é que uma string PHP que represente uma expressão regular contendo uma barra invertida precisa de quatro barras invertidas. O interpretador do PHP analisará as quatro barras invertidas como duas. Então, o interpretador de expressões regulares analisará as duas como uma.

O cifrão também é um caractere especial em strings com aspas duplas no PHP e em expressões regulares. Para inserir um $ literal em um padrão, você precisa de "\\\$". Como essa string está entre aspas duplas, o PHP a analisará como \$, que o interpretador de expressões regulares pode então reconhecer como o cifrão.

Resumo de caracteres especiais

Um resumo de todos os caracteres especiais é mostrado nas Tabelas 4.4 e 4.5. A Tabela 4.4 mostra o significado de caracteres especiais fora de colchetes e a Tabela 4.5 mostra seu significado quando utilizados dentro de colchetes.

Tabela 4.4 **Resumo de caracteres especiais utilizados em expressões regulares do POSIX fora dos colchetes**

Caractere	Significado
\	Caractere de escape.
^	Procura uma correspondência no início da string.
$	Procura uma correspondência no final da string.
.	Identifica qualquer caractere, exceto nova linha (\n).
|	Início da ramificação alternativa (leitura como OR).
(	Inicia subpadrão.
)	Termina subpadrão.
*	Repete 0 ou mais vezes.
+	Repete 1 ou mais vezes.
{	Inicia o quantificador min/max.
}	Termina o quantificador min/max.
?	Marca um subpadrão como opcional.

Tabela 4.5 **Resumo de caracteres especiais utilizados em expressões regulares do POSIX entre colchetes**

Caractere	Significado
\	Caractere de escape.
^	NÃO, somente se utilizado na posição inicial.
-	Utilizado para especificar intervalos de caractere.

Juntando tudo no Smart Form

Há pelo menos duas possíveis utilizações de expressões regulares na aplicação Smart Form. A primeira utilização é detectar termos particulares no feedback do cliente. Podemos ser um pouco mais inteligentes em relação a isso utilizando expressões regulares. Utilizando uma função de string, teríamos de fazer três pesquisas diferentes se quiséssemos localizar `'shop'`, `'customer service'` ou `'retail'`. Com uma expressão regular, podemos localizar todos os três:

```
shop|customer service|retail
```

A segunda utilização é validar endereços de e-mail de cliente na nossa aplicação codificando o formato padronizado de um endereço de e-mail em uma expressão regular. O formato inclui alguns caracteres alfanuméricos de pontuação, seguido por um símbolo `@`, seguido por uma string de caracteres alfanuméricos e hífen, seguido por um ponto, seguido por mais caracteres de hífen e alfanuméricos possivelmente mais pontos, até o final da string, que codifica da seguinte maneira:

```
^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$
```

A subexpressão `^[a-zA-Z0-9_\-\.]+` significa "inicie a string com pelo menos uma letra, número, sublinhado, hífen, ponto ou alguma combinação disso".

O símbolo `@` localiza um literal `@`.

A subexpressão `[a-zA-Z0-9\-]+` localiza a primeira parte do nome de host incluindo caracteres alfanuméricos e hifens. Observe que cortamos o hífen porque ele é um caractere especial entre colchetes.

A combinação `\.` identifica um ponto literal (.).

A subexpressão `[a-zA-Z0-9\-\.]+$` localiza o restante de um nome de domínio, incluindo letras, números, hifens e mais pontos se requerido, até o final da string.

Um pouco de análise mostra que você pode produzir endereços inválidos de e-mail que ainda serão identificados por essa expressão regular. É quase impossível capturar todos eles, mas isso melhorará um pouco a situação. Você pode refinar essa expressão de várias maneiras. Você pode, por exemplo, listar os TLDs válidos. Tenha cuidado ao tornar as coisas mais restritivas, uma vez que uma função de validação que rejeita 1% dos dados válidos é muito mais irritante que uma que permite 10% de dados inválidos.

Agora que você leu sobre expressões regulares, veremos as funções do PHP que as utilizam.

Localizando substrings com expressões regulares

Localizar substrings é a principal aplicação das expressões regulares que acabamos de desenvolver. As duas funções disponíveis no PHP para localizar expressões regulares são `ereg( )` e `eregi( )`. A função `ereg( )` tem o seguinte protótipo:

```
int ereg(string padrão, string pesquisa, array [correspondência]);
```

Essa função pesquisa a string *pesquisa*, procurando correspondências com a expressão regular em *padrão*. Se forem localizadas correspondências para subexpressões de *padrão*, elas serão armazenadas no array *correspondências*, uma subexpressão por elemento do array.

A função `eregi( )` é idêntica exceto que não faz distinção de letras maiúsculas e minúsculas.

Podemos adaptar o exemplo Smart Form para utilizar expressões regulares da seguinte maneira:

```
if (!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$', $email))
{
  echo 'That is not a valid email address.  Please return to the'
      .' previous page and try again.';
  exit;
}
$toaddress = 'feedback@example.com';  // o valor padrão
if (eregi('shop|customer service|retail', $feedback))
  $toaddress = 'retail@example.com';
else if (eregi('deliver.*|fulfil.*', $feedback))
  $toaddress = 'fulfilment@example.com';
else if (eregi('bill|account', $feedback))
  $toaddress = 'accounts@example.com';

if (eregi('bigcustomer\.com', $email))
    $toaddress = 'bob@example.com';
```

Substituindo substrings por expressões regulares

Você também pode utilizar expressões regulares para localizar e substituir substrings da mesma maneira que utilizamos `str_replace( )`. As duas funções disponíveis para isso são `ereg_replace( )` e `eregi_replace( )`. A função `ereg_replace( )` tem o seguinte protótipo:

```
string ereg_replace(string padrão, string substituta, string pesquisa);
```

Essa função procura a expressão regular *padrão* na string *pesquisa* e a substitui pela string *substituta*.

A função `eregi_replace( )` é idêntica, mas novamente, não faz distinção entre letras maiúsculas e minúsculas.

Dividindo strings com expressões regulares

Outra função de expressão regular útil é split( ), que tem o seguinte protótipo:

```
array split(string padrão, string pesquisa, int [máx]);
```

Essa função divide a string *pesquisa* em substrings na expressão regular *padrão* e retorna as substrings em um array. O inteiro *máx* limita o número de itens que podem entrar no array.

Isso pode ser útil para dividir nomes de domínio ou datas. Por exemplo,

```
$domain = 'yallara.cs.rmit.edu.au';
$arr = split ('\.', $domain);
while (list($key, $value) = each ($arr))
  echo '<br />'.$value;
```

Isso divide o nome de host em seus cinco componentes e imprime cada um deles em uma linha separada.

Comparação de funções de string e funções de expressão regular

Em geral, as funções de expressão regular executam com menos eficiência do que as funções de string com funcionalidade semelhante. Se sua aplicação for suficientemente simples para utilizar expressões de string, faça isso. Isso pode não ser verdadeiro para tarefas que possam ser desempenhadas com uma única expressão regular mas sim com diversas funções de string.

Leitura adicional

PHP tem muitas funções de string. Abrangemos as mais úteis neste capítulo, mas se você tiver uma necessidade particular (como converter caracteres para o cirílico), examine o manual on-line do PHP para verificar se o PHP tem a função para você.

A quantidade de material disponível nas expressões regulares é enorme. Você pode iniciar com a página man para regexp se estiver utilizando UNIX e há também alguns artigos impressionantes em devshed.com e phpbuilder.com.

No Web site da Zend, você pode ver uma função de validação de e-mail poderosa e mais complexa que a função que desenvolvemos aqui. Chama-se MailVal( ) e está disponível em http://www.zend.com/codex.php?id=88&single=1.

Expressões regulares exigem um pouco de tempo para serem dominadas – quanto maior o número de exemplos que você estuda e executa, mais confiança você terá em utilizá-las.

A seguir

No próximo capítulo, discutiremos várias maneiras de utilizar o PHP a fim de economizar tempo e esforço de programação e impedir a redundância reutilizando código preexistente.

5

Reutilizando código e escrevendo funções

ESTE CAPÍTULO EXPLICA COMO A REUTILIZAÇÃO DE CÓDIGO resulta em código mais consistente, mais fácil de fazer manutenção e mais confiável com menos esforço. Demonstraremos técnicas para modularizar e reutilizar código, começando com a utilização simples de `require( )` e `include( )` para utilizar o mesmo código em mais de uma página. Explicaremos por que essas funções são superiores aos includes do lado servidor. O exemplo dado abrangerá a utilização de arquivos includes para obter aparência e comportamento consistentes por todo o site. Explicaremos como escrever e chamar suas próprias funções utilizando funções de geração de formulários e de geração de páginas como exemplos.

Neste capítulo, abordaremos:

- Reutilizando o código;
- Utilizando `require( )` e `include( )`;
- Introduzindo funções;
- Definindo funções;
- Usando parâmetros;
- Compreendendo escopo;
- Retornando valores;
- Chamada por referência *versus* chamada por valor;
- Implementando recursão.

Reutilizando código

Um dos objetivos dos engenheiros de software é reutilizar código em vez de escrever código novo. A razão disso não é que os engenheiros de software sejam um grupo particularmente preguiçoso. Reutilizar código existente reduz custos, aumenta a confiabilidade e melhora a consistência. De modo ideal, um novo projeto é criado combinando componentes reutilizáveis existentes, com um mínimo de desenvolvimento a partir do zero.

Custo

Durante a vida útil de uma parte de software, mais tempo será significativamente gasto mantendo, modificando, testando e documentando do que foi originalmente gasto escrevendo-o. Se estiver es-

crevendo código comercial, você deveria tentar limitar o número de linhas que estão em uso dentro da organização. Uma das maneiras mais práticas de alcançar isso é reutilizar código já em uso em vez de escrever uma versão ligeiramente diferente do mesmo código para uma nova tarefa. Menos código significa menos custo. Se existir software que atenda às exigências do novo projeto, adquira-o. O custo de comprar software existente é quase sempre menor que o custo de desenvolver um produto equivalente. No entanto, seja prudente se houver um software que quase atenda aos seus requisitos. Pode ser mais difícil modificar código existente que escrever novo código.

Confiabilidade

Se um módulo de código estiver em utilização em algum lugar da sua organização, presumivelmente ele já foi completamente testado. Mesmo que ele tenha apenas algumas linhas, há uma possibilidade de que, se reescrevê-lo, você deixe passar algo que o autor original incorporou ou algo que foi adicionado ao código original depois que um defeito foi localizado durante o teste. O código existente e maduro (completamente desenvolvido) é normalmente mais confiável que um código fresco e "verde".

Consistência

As interfaces externas para seu sistema, incluindo tanto interfaces com o usuário como interfaces fora de sistemas, devem ser consistentes. Exige vontade e um esforço deliberado escrever um novo código que seja consistente com a maneira como as outras partes do sistema funcionam. Se estiver reutilizando código que executa outra parte do sistema, sua funcionalidade deve ser automaticamente consistente.

Acima de todas essas vantagens, a reutilização do código representa menos trabalho para você, se o código original for modular e bem escrito. Enquanto trabalha, tente reconhecer as seções de código que você talvez seja capaz de chamar novamente no futuro.

Utilizando `require( )` e `include( )`

O PHP fornece duas instruções muito simples e ainda muito úteis para permitir reutilizar qualquer tipo de código. Utilizando uma instrução `require( )` ou `include( )`, você pode carregar um arquivo no script de PHP. O arquivo pode conter qualquer coisa que você normalmente digitaria em um script incluindo instruções de PHP, texto, tags HTML, funções de PHP ou classes de PHP.

Essas instruções funcionam de maneira semelhante ao Server Side Includes oferecido por muitos servidores Web e instruções `#include` em C ou C++.

Utilizando `require( )`

O código a seguir é armazenado em um arquivo chamado `reusable.php`:

```
<?php
  echo 'Here is a very simple PHP statement.<br />';
?>
```

O código a seguir é armazenado em um arquivo chamado `main.php`:

```
<?php
  echo 'This is the main file.<br />';
  require( 'reusable.php' );
  echo 'The script will end now.<br />';
?>
```

Se você carregar `reusable.php`, esse arquivo provavelmente não o surpreenderá quando *`Here is a very simple PHP statement`* aparecer no navegador. Se carregar `main.php`, algo um pouco mais interessante acontece. A saída desse script é mostrada na Figura 5.1.

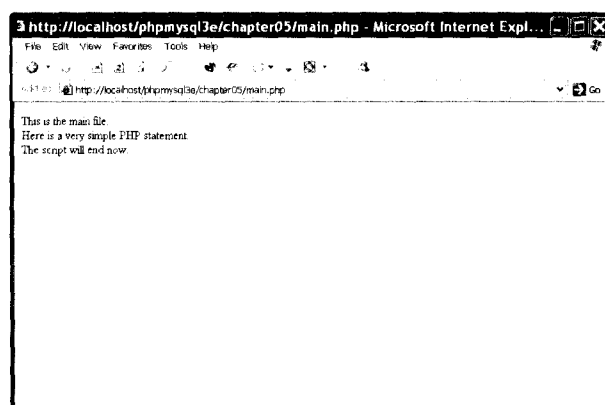



Figura 5.1 A saída de `main.php` mostra o resultado da instrução `require( )`.

É necessário um arquivo para utilizar uma instrução `require( )`. No exemplo anterior, estamos utilizando o nome de arquivo `reusable.php`. Quando executamos nosso script, a instrução `require( )`

```
require( 'reusable.php' );
```

é substituída pelo conteúdo do arquivo solicitado e o script então é executado. Isso significa que quando carregamos `main.php`, este executa como se o script fosse escrito da seguinte maneira:

```
<?php
  echo 'This is the main file.<br />';
  echo 'Here is a very simple PHP statement.<br />';
  echo 'The script will end now.<br />';
?>
```

Ao utilizar `require( )` você precisa notar as diferentes maneiras como as extensões do nome do arquivo e as tags de PHP são tratadas.

Extensões de nome de arquivo e require( )

O PHP não vê a extensão do nome do arquivo no arquivo requerido. Isso significa que você pode nomear seu arquivo como quiser contanto que não vá chamá-lo diretamente. Quando você utilizar `require( )` para carregar o arquivo, ele efetivamente se tornará parte de um arquivo de PHP e será executado como tal.

Normalmente, as instruções do PHP não seriam processadas se estivessem em um arquivo denominado, por exemplo, `page.html`. O PHP normalmente só é chamado para analisar sintaticamente arquivos com extensões definidas como `.php`. Entretanto, se você carregar essa `page.html` via uma instrução `require( )`, qualquer PHP dentro dela será processado. Portanto, você pode utilizar a extensão que preferir para incluir arquivos, mas seria uma boa idéia tentar ater-se a uma convenção sensata, como `.inc` ou `.php`.

Algo de que se deve estar ciente é que se arquivos terminando em `.inc` ou alguma outra extensão não-padrão são armazenados na árvore de documentos Web e os usuários os carregam diretamente no navegador, eles serão capazes de ver o código em texto simples, incluindo qualquer senha. Portanto, é importante armazenar arquivos incluídos fora da árvore de documentos ou utilizar as extensões padrão.

Tags de PHP e require( )

No exemplo, nosso arquivo reutilizável (`reusable.php`) foi escrito da seguinte maneira:

```
<?php
  echo 'Here is a very simple PHP statement.<br />';
?>
```

Colocamos o código de PHP dentro do arquivo nas tags de PHP. Você precisará fazer isso se quiser que o código de PHP dentro de um arquivo requerido seja tratado como código PHP. Se você não abrir uma tag de PHP, seu código será tratado somente como texto ou HTML e não será executado.

*Utilizando **require( )** para modelos de Web site

Se sua empresa tiver uma aparência e um comportamento consistentes nas páginas do Web site, você pode utilizar o PHP para adicionar o modelo e elementos padrão às páginas utilizando `require( )`.

Por exemplo, o Web site da empresa fictícia TLA Consulting tem várias páginas, todas elas com a aparência e o comportamento mostrados na Figura 5.2. Quando for necessária uma nova página, o desenvolvedor pode abrir uma página existente, recortar o texto existente do meio do arquivo, inserir novo texto e salvar o arquivo sob um novo nome.

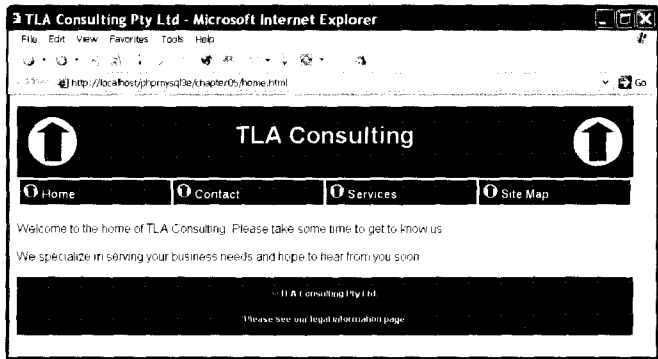



Figura 5.2 A TLA Consulting tem aparência e comportamento padrão para todas as suas páginas Web.

Considere este cenário: o Web site existe há pouco tempo e há agora dezenas, centenas ou talvez até milhares de páginas, todas elas seguindo um estilo comum. Então, toma-se a decisão de alterar parte da aparência padrão – isso poderia ser algo menor, como adicionar um endereço de e-mail ao rodapé de cada página ou adicionar uma nova única entrada ao menu de navegação. Você quer fazer essa pequena alteração em dezenas, centenas ou até em milhares de páginas?

Reutilizar diretamente as seções de HTML que são comuns a todas as páginas é uma abordagem muito melhor que recortar e colar em dezenas, centenas ou até milhares de páginas. O código-fonte para a homepage (`home.html`) mostrado na Figura 5.2 é fornecido na Listagem 5.1.

Listagem 5.1 **home.html** – **A HTML que produz a homepage da TLA Consulting**

```
<html>
<head>
  <title>TLA Consulting Pty Ltd</title>
  <style>
    h1 {color:white; font-size:24pt; text-align:center;
        font-family:arial,sans-serif}
    .menu {color:white; font-size:12pt; text-align:center;
           font-family:arial,sans-serif; font-weight:bold}
    td {background:black}
    p {color:black; font-size:12pt; text-align:justify;
       font-family:arial,sans-serif}
    p.foot {color:white; font-size:9pt; text-align:center;
            font-family:arial,sans-serif; font-weight:bold}
    a:link,a:visited,a:active {color:white}
  </style>
```

Listagem 5.1 **Continuação**

```
</head>
<body>

  --> page header -->
  <table width="100%" cellpadding="12" cellspacing="0" border="0">
  <tr bgcolor="black">
    <td align="left"><img src="logo.gif"></td>
    <td>
        <h1>TLA Consulting</h1>
    </td>
    <td align="right"><img src="logo.gif"></td>
  </tr>
  </table>

  <!-- menu -->
  <table width="100%" bgcolor="white" cellpadding="4" cellspacing="4">
  <tr >
    <td width="25%">
      <img src="s-logo.gif"> <span class="menu">Home</span></td>
    <td width="25%">
      <img src="s-logo.gif"> <span class="menu">Contact</span></td>
    <td width="25%">
      <img src="s-logo.gif"> <span class="menu">Services</span></td>
    <td width="25%">
      <img src="s-logo.gif"> <span class="menu">Site Map</span></td>
  </tr>
  </table>

  <!-- page content -->
  <p>Welcome to the home of TLA Consulting.
  Please take some time to get to know us.</p>
  <p>We specialize in serving your business needs
  and hope to hear from you soon.</p>

  <!-- page footer -->
  <table width="100%" bgcolor="black" cellpadding="12" border="0">
  <tr>
    <td>
      <p class="foot">&copy; TLA Consulting Pty Ltd.</p>
      <p class="foot">Please see our <a href="válida.php">válida information page</a></p>
    </td>
  </tr>
  </table>
</body>
</html>
```

Você pode ver na Listagem 5.1 que existem várias seções distintas de código nesse arquivo. O cabeçalho HTML contém definições de folha de estilo em cascata (Cascading Style Sheet – CSS) utilizadas pela página. A seção rotulada "página de cabeçalho" exibe o nome da empresa e o logotipo, a "barra de menu" cria a barra de navegação da página e "conteúdo da página" é texto único para essa página. Abaixo disso, está o rodapé de página. Podemos dividir utilmente esse arquivo e nomear as partes `header.inc`, `home.php` e `footer.inc`. `header.inc` e `footer.inc` contêm o código que será reutilizado em outras páginas.

O arquivo `home.php` é um substituto da página `home.html` e contém o conteúdo exclusivo da página e duas instruções `require( )` como mostrado na Listagem 5.2.

Listagem 5.2 **home.php – O PHP que produz a homepage da TLA Consulting**

```
<?php
  require('header.inc');
?>
  <!-- page content -->
  <p>Welcome to the home of TLA Consulting.
  Please take some time to get to know us.</p>
  <p>We specialize in serving your business needs
  and hope to hear from you soon.</p>
<?php
  require('footer.inc');
?>
```

As instruções `require( )` na `home.php` carregam `header.inc` e `footer.inc`.

Como mencionado, o nome dado a esses arquivos não afeta a maneira como eles são processados quando os chamamos via `require( )`. Uma convenção comum, mas inteiramente opcional, é chamar os arquivos parciais que terminarão incluídos em outros arquivos *algumacoisa*.inc (aqui `inc` significa include). Também é comum, e uma boa idéia, colocar os arquivos include em um diretório que possa ser visto pelos scripts, mas não permitir que os arquivos include sejam carregados individualmente via servidor Web. Isso impedirá que esses arquivos sejam carregados individualmente, o que a) provavelmente produzirá alguns erros se a extensão de arquivo for `.php` mas contiver somente uma página parcial ou script; ou b) permitirá que as pessoas leiam seu código-fonte se você tiver utilizado outra extensão.

O arquivo `header.inc` contém as definições de CSS que a página utiliza, as tabelas que exibem o nome de empresa e os menus de navegação como mostra a Listagem 5.3.

Listagem 5.3 **header.inc – O cabeçalho reutilizável para todas as páginas Web da TLA**

```
<html>
<head>
  <title>TLA Consulting Pty Ltd</title>
  <style>
    h1 {color:white; font-size:24pt; text-align:center;
        font-family:arial,sans-serif}
    .menu {color:white; font-size:12pt; text-align:center;
           font-family:arial,sans-serif; font-weight:bold}
    td {background:black}
    p {color:black; font-size:12pt; text-align:justify;
      font-family:arial,sans-serif}
    p.foot {color:white; font-size:9pt; text-align:center;
            font-family:arial,sans-serif; font-weight:bold}
    a:link,a:visited,a:active {color:white}
  </style>
</head>
<body>

  <!-- page header -->
  <table width="100%" cellpadding="12" cellspacing="0" border="0">
  <tr bgcolor="black">
    <td align="left"><img src="logo.gif"></td>
    <td>
        <h1>TLA Consulting</h1>
    </td>
    <td align="right"><img src="logo.gif"></td>
  </tr>
</table>
```

Listagem 5.3 **Continuação**

```
<!-- menu -->
<table width="100%" bgcolor="white" cellpadding="4" cellspacing="4">
<tr >
  <td width="25%">
    <img src="s-logo.gif"> <span class="menu">Home</span></td>
  <td width="25%">
    <img src="s-logo.gif"> <span class="menu">Contact</span></td>
  <td width="25%">
    <img src="s-logo.gif"> <span class="menu">Services</span></td>
  <td width="25%">
    <img src="s-logo.gif"> <span class="menu">Site Map</span></td>
</tr>
</table>
```

O arquivo `footer.inc` contém a tabela que exibe o rodapé na parte inferior de cada página. Esse arquivo é mostrado na Listagem 5.4.

Listagem 5.4 **`footer.inc` – O rodapé reutilizável para todas as páginas Web da TLA**

```
<!-- page footer -->
  <table width="100%" bgcolor="black" cellpadding="12" border="0">
  <tr>
    <td>
      <p class="foot">&copy; TLA Consulting Pty Ltd.</p>
      <p class="foot">Please see our
                   <a href="válida.php">válida information page</a></p>
    </td>
  </tr>
  </table>
</body>
</html>
```

Essa abordagem fornece muito facilmente um Web site de aparência consistente e você pode fazer uma nova página no mesmo estilo digitando algo assim:

```
<?php require('header.inc'); ?>
Here is the content for this page
<?php require('footer.inc'); ?>
```

O mais importante é que mesmo depois de criarmos muitas páginas utilizando esse cabeçalho é fácil alterar os arquivos de cabeçalho e rodapé. Quer você faça uma alteração de texto menor ou reprojete completamente o visual do site, só precisa fazer a alteração uma vez. Não precisamos alterar separadamente cada página no site porque todas elas estão carregando os arquivos de cabeçalho e rodapé.

O exemplo mostrado aqui utiliza somente HTML simples no corpo, cabeçalho e rodapé. Esse não precisa ser o caso. Dentro desses arquivos, poderíamos utilizar instruções de PHP para gerar dinamicamente partes da página.

Se quiser ter certeza de que um arquivo será tratado como texto simples ou HTML, e não terá nenhum PHP executado, utilize então `readfile( )`. Essa função ecoa o conteúdo de um arquivo sem analisá-lo. Essa pode ser uma precaução importante de segurança se você estiver utilizando texto fornecido pelo usuário.

Utilizando `include( )`

As instruções `require( )` e `include( )` são quase idênticas. A única diferença entre elas é que quando falham, `require( )` retorna um erro fatal, enquanto `include( )` retorna apenas um aviso.

Utilizando require_once( ) e include_once( )

Existem duas variações de require( ) e include( ), chamadas require_once( ) e include_once( ), respectivamente. O objetivo dessas construções, como você poderia supor, é garantir que um arquivo possa ser incluído apenas uma vez (*once*). Para os exemplos vistos até aqui – com cabeçalhos e rodapés – essa funcionalidade não é particularmente útil.

Ela se torna útil quando você começa a usar require( ) e include( ) para incluir bibliotecas de funções. Usar essas construções oferece uma proteção para você não incluir acidentalmente a mesma biblioteca de função duas vezes, e, portanto, redefinindo funções e gerando um erro.

Utilizando auto_prepend_file e auto_append_file

Se quiser utilizar require( ) ou include( ), adicionar cabeçalho e rodapé a cada página, pode fazê-lo de outra maneira. Duas das opções de configuração no arquivo php.ini são auto_prepend_file e auto_append_file. Definindo essas opções de modo a apontar para os arquivos de cabeçalho e rodapé, você garante que eles serão carregados antes e depois de cada página. Os arquivos incluídos com essas diretivas se comportam como se tivessem sido adicionados usando uma instrução include( ); ou seja, se o arquivo estiver faltando, surgirá um aviso.

Para o Windows, as configurações são semelhantes a:

```
auto_prepend_file = "c:/Apache/include/header.inc"
auto_append_file = "c:/Apache/include/footer.inc"
```

Para o Unix, se parecem com:

```
auto_prepend_file = "/home/username/include/header.inc"
auto_append_file = "/home/username/include/footer.inc"
```

Se você utilizar essas diretivas, não precisa digitar as instruções include( ), mas os cabeçalhos e rodapés não serão mais opcionais nas páginas.

Se você está utilizando um servidor Web Apache, pode mudar diversas opções de configuração como essas para os diretórios individuais. Para isso, é preciso que o servidor esteja definido para aceitar que o(s) arquivo(s) de configuração principal(is) seja(m) sobrescrito(s). Para definir *auto prepend* e *auto append* para um diretório, crie um arquivo chamado .htaccess no diretório. O arquivo precisa conter estas duas linhas:

```
php_value auto_prepend_file "/home/username/include/header.inc"
php_value auto_append_file "/home/username/include/footer.inc"
```

Observe que a sintaxe é um pouco diferente da mesma opção em php.ini: assim como php_value no início da linha, não há sinal de igual. Várias outras definições de configuração php.ini também podem ser alteradas dessa maneira.

Configurar as opções no arquivo .htaccess, em vez de em php.ini ou no arquivo de configuração do seu servidor Web, oferece muito mais flexibilidade. Você pode alterar as configurações em uma máquina compartilhada que afetem apenas os seus diretórios. Não é preciso reiniciar o servidor Web, e você não precisa de acesso de administrador. Uma desvantagem ao método .htaccess é que os arquivos são lidos e analisados a cada vez que é requerido um arquivo naquele diretório, em vez de apenas uma vez na inicialização; portanto, há uma queda do desempenho.

Utilizando funções no PHP

As funções existem na maioria das linguagens de programação. Elas são utilizadas para separar código que realiza uma tarefa única e bem-definida. Isso torna o código mais fácil de ler e permite reutilizar o código toda vez que precisarmos fazer a mesma tarefa.

Uma função é um módulo autocontido de código que prescreve uma interface de chamada, realiza alguma tarefa e opcionalmente retorna um resultado.

Você já viu várias funções. Nos capítulos precedentes, rotineiramente chamamos várias funções que vêm predefinidas no PHP. Também escrevemos algumas funções simples mas ocultamos os detalhes. Nesta seção, abordaremos como chamar e escrever funções em mais detalhe.

Chamando funções

A próxima linha é a chamada mais simples possível a uma função:

```
nome_da_função( );
```

Isso chama uma função denominada `nome_da_função` que não requer parâmetros. Essa linha de código ignora qualquer valor que talvez possa ser retornado por essa função.

Várias funções são chamadas exatamente dessa maneira. A função `phpinfo( )` é freqüentemente útil para fazer testes porque exibe a versão de PHP instalada, informações sobre o PHP, a configuração do servidor Web e os valores de diversas variáveis do PHP e variáveis de servidor. Essa função não aceita nenhum parâmetro e geralmente ignoramos seu valor de retorno, então uma chamada a `phpinfo( )` será semelhante ao seguinte:

```
phpinfo( );
```

A maioria das funções requer um ou mais parâmetros – informações fornecidas a uma função quando elas são chamadas, informações essas que influenciam o resultado da execução da função. Passamos parâmetros colocando os dados ou o nome de uma variável que armazena os dados entre parênteses depois do nome da função. Uma chamada a uma função com um parâmetro se assemelha ao seguinte:

```
nome_da_função('parâmetro');
```

Nesse caso, o parâmetro que utilizamos foi uma string contendo apenas a palavra parâmetro, mas as chamadas a seguir também são válidas dependendo da função:

```
nome_da_função(2);
nome_da_função(7.993);
nome_da_função($variavel);
```

Na última linha, `$variável` poderia ser qualquer tipo de variável de PHP, incluindo um array.

Um parâmetro pode ser qualquer tipo de dados, mas funções particulares normalmente irão requerer tipos de dados particulares.

Você pode ver quantos parâmetros uma função assume, o que cada um representa e que tipo de dados cada um precisa para ser do *protótipo* da função. Freqüentemente mostramos o protótipo quando descrevemos uma função.

Esse é protótipo para a função `fopen( )`:

```
resource fopen( string nome_do_arquivo, string modo,
                [, bool usar_caminho_do_include [, resource zcontext]] );
```

O protótipo fornece várias informações e é importante que você saiba como interpretar corretamente essas especificações. Nesse caso, a palavra resource antes do nome da função diz que essa função retornará um recurso (ou seja, um handle de arquivo aberto). Os parâmetros de função estão dentro dos parênteses. No caso de `fopen( )`, quatro parâmetros são mostrados no protótipo. O parâmetro *nome_do_arquivo* e o *modo* são strings, o parâmetro *usar_caminho_do_include* é um booleano e o parâmetro `zcontext` é um recurso. Os colchetes que cercam *usar_caminho_do_include* e `zcontext` indicam que esses parâmetros são opcionais. Podemos fornecer valores para parâmetros opcionais ou escolher ignorá-los, e o valor padrão será utilizado. Observe, entretanto, que para uma função com mais de um parâmetro opcional, só é possível omitir parâmetros a partir da direita. Por exem-

plo, ao usar fopen( ), você pode omitir zcontext ou tanto zcontext como *usar_caminho_do_include*; entretanto, você não pode omitir *usar_caminho_do_include* e fornecer zcontext.

Depois de ler o protótipo para essa função, sabemos que o seguinte fragmento de código será uma chamada válida para fopen( ):

```
$name = 'myfile.txt';
$openmode = 'r';
$fp = fopen($name, $openmode)
```

Esse código chama a função denominada fopen( ). O valor retornado pela função será armazenado na variável $fp. Escolhemos passar para a função uma variável denominada $name que contém uma string representando o arquivo que queremos abrir e uma variável denominada $openmode contendo uma string representando o modo como queremos abrir o arquivo. Escolhemos não fornecer o terceiro parâmetro opcional.

Chamada para uma função indefinida

Se tentar chamar uma função que não existe, você obterá uma mensagem de erro como mostrado na Figura 5.3.

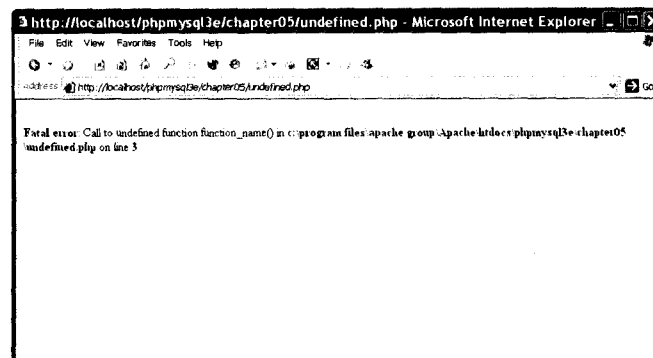



Figura 5.3 Essa mensagem de erro é o resultado de chamar uma função que não existe.

As mensagens de erro que o PHP fornece são normalmente muito úteis. Essa informa exatamente em qual arquivo o erro ocorreu, em qual linha do script ele ocorreu e o nome da função que tentamos chamar. Isso deve tornar relativamente fácil a localização e correção.

Se vir essa mensagem de erro, você deve verificar:

- O nome de função foi escrito corretamente?
- A função existe na versão de PHP que você está utilizando?

Nem sempre é fácil lembrar como um nome de função é escrito. Por exemplo, alguns nomes de função com duas palavras têm um sublinhado entre as palavras e alguns não. A função stripslashes( ) junta duas palavras, enquanto a função strip_tags( ) separa as palavras com um sublinhado. Escrever errado o nome de uma função em uma chamada de função resulta em um erro como mostra a Figura 5.3.

Muitas funções utilizadas neste livro não existem no PHP 3.0 porque este livro assume que você está utilizando pelo menos o PHP 4.0. Em cada nova versão, novas funções são definidas e se você estiver utilizando uma versão mais antiga, as funcionalidades e o desempenho adicionados justificam uma atualização. Para ver quando uma função particular foi adicionada, você pode verificar o manual on-line. Tentar chamar uma função não declarada na versão que está executando resultará em um erro como o mostrado na Figura 5.3.

Uma outra razão pela qual você pode ver essa mensagem de erro é quando a função que está chamando faz parte de uma extensão PHP que não esteja carregada. Por exemplo, se você tentar utilizar funções da biblioteca gd (manipulação de imagens) e não tiver instalado-a, verá essa mensagem.

Entendendo maiúsculas e minúsculas e nomes de função

Observe que as chamadas para funções *não* fazem distinção entre letras maiúsculas e minúsculas, assim `nome_da_função( )`, `Nome_da_função( )` ou `NOME_DA_FUNÇÃO( )` são todas maneiras válidas de chamar e terão o mesmo resultado. Você é livre para usar maiúsculas e minúsculas da maneira como achar mais fácil de ler, mas deve atentar para a consistência. A convenção utilizada neste livro, e na maior parte da documentação do PHP, é usar todas as letras em minúsculas.

É importante notar que nomes de função comportam-se de maneira diferente de variáveis. Os nomes de variável *distinguem* letras maiúsculas de minúsculas, então `$Name` e `$name` são duas variáveis separadas, mas `Name( )` e `name( )` são a mesma função.

Por que você deve definir suas próprias funções?

Nos capítulos precedentes, você viu muitos exemplos utilizando algumas das funções predefinidas do PHP. Entretanto, o verdadeiro poder de uma linguagem de programação vem da capacidade de criar suas próprias funções.

As funções predefinidas do PHP permitem interagir com arquivos, utilizar um banco de dados, criar imagens gráficas e conectar-se a outros servidores. Entretanto, na sua carreira há muitas ocasiões em que você precisará fazer algo que os criadores da linguagem não previram.

Felizmente, você não está limitado à utilização das funções predefinidas porque pode escrever suas próprias funções para realizar a tarefa que quiser. Provavelmente seu código será uma mistura de funções existentes combinadas com sua própria lógica para realizar uma tarefa para você. Se estiver escrevendo um bloco de código para uma tarefa que você possivelmente vai querer reutilizar em vários lugares em um script ou em vários scripts, seria inteligente declarar esse bloco como uma função.

Declarar uma função permite utilizar seu próprio código da mesma maneira como as funções predefinidas. Você simplesmente chama sua função e fornece a ela os parâmetros necessários. Isso significa que você pode chamar e reutilizar a mesma função muitas vezes por todo seu script.

Examinando a estrutura básica da função

Uma declaração de função cria ou *declara* uma nova função. A declaração começa com a palavra-chave `function`, fornece o nome de função, os parâmetros necessários e contém o código que será executado toda vez que essa função for chamada.

Eis a declaração de uma função trivial:

```
function my_function( )
{
  echo 'My function was called';
}
```

Essa declaração de função inicia com `function`, de modo que os leitores humanos e o analisador de sintaxe do PHP saibam que o que segue será uma função definida pelo usuário. O nome de função é `my_function`. Podemos chamar nossa nova função com a instrução a seguir:

```
my_function( );
```

Como você provavelmente adivinhou, chamar essa função resultará no texto `"My function was called"` que aparecer no navegador do visualizador.

As funções predefinidas estão disponíveis para todos os scripts de PHP, mas se você declarar suas próprias funções, elas só estão disponíveis para o(s) script(s) em que foram declaradas. É uma

boa idéia ter um arquivo contendo suas funções comumente utilizadas. Então, você pode ter uma instrução `require( )` em todos os scripts para tornar as funções disponíveis.

Dentro de uma função, as chaves incluem o código que realiza a tarefa exigida. Entre essas chaves, você pode ter qualquer coisa que seja válida em outra parte de um script de PHP incluindo chamadas de função, declarações de novas variáveis ou funções, instruções `require( )` ou `include( )` e HTML simples. Se quisermos sair do PHP dentro de uma função e tipo HTML simples, fazemos isso da mesma maneira como faríamos em qualquer outro lugar no script – com uma tag PHP de fechamento seguida pela HTML. A seguir é mostrada uma modificação válida do exemplo anterior que produz a mesma saída:

```
<?php
  function my_function( )
  {
?>
My function was called
<?php
  }
?>
```

Observe que o código de PHP está incluído dentro das tags PHP de abertura e fechamento correspondentes. Para a maioria dos fragmentos de código de exemplo neste livro, não mostramos essas tags. Elas são mostrados aqui porque são exigidas dentro do exemplo bem como acima e abaixo dele.

Nomeando a função

O mais importante a considerar ao nomear funções é que o nome deve ser curto mas descritivo. Se a função cria um cabeçalho de página, `pageheader( )` ou `page_header( )` podem ser bons nomes.

Algumas restrições são:

- A função não pode ter o mesmo nome que uma função existente.
- O nome de função somente pode conter letras, dígitos e sublinhados.
- O nome de função não pode iniciar com um dígito.

Muitas linguagens realmente permitem reutilizar nomes de função. Esse recurso é chamado de *sobrecarga de funções*. Entretanto, o PHP não suporta sobrecarga de funções, então sua função não pode ter o mesmo nome que qualquer função predefinida ou uma função existente definida pelo usuário. Observe que embora cada script de PHP conheça todas as funções predefinidas, as funções definidas pelo usuário só existem nos scripts em que são declaradas. Isso significa que você poderia reutilizar um nome de função em um arquivo diferente, mas isso resultaria em confusão e deve ser evitado.

Os seguintes nomes de função são válidos:

```
name( )
name2( )
name_three( )
_namefour( )
```

Esses são inválidos:

```
5name( )
name-six( )
fopen( )
```

(O último nome seria válido se já não existisse.)

Observe que, embora `$name` não seja um nome válido para uma função, uma chamada de função do tipo

```
$name( );
```

pode executar bem, dependendo do valor de $name. Isso porque o PHP pega o valor armazenado em $name, procura por uma função com aquele nome e tenta chamá-la para você. Esse tipo de função é conhecido como *função de variável* e pode ser útil para você ocasionalmente.

Usando parâmetros

Para fazer seu trabalho, a maioria das funções requer um ou mais parâmetros. Um parâmetro permite passar dados para uma função. Eis um exemplo de uma função que requer um parâmetro. Essa função aceita um array dimensional e o exibe como uma tabela.

```
function create_table($data)
{
  echo '<table border="1">';
  reset($data); // Lembra de que isso é utilizado para apontar para o início
  $value = current($data);
  while ($value)
  {
     echo "<tr><td>$value</td></tr>\n";
     $value = next($data);
  }
  echo '</table>';
}
```

Se chamarmos a função create_table( ) da seguinte maneira:

```
$my_array = array('Line one.','Line two.','Line three.');
create_table($my_array);
```

veremos uma saída semelhante à mostrada na Figura 5.4.

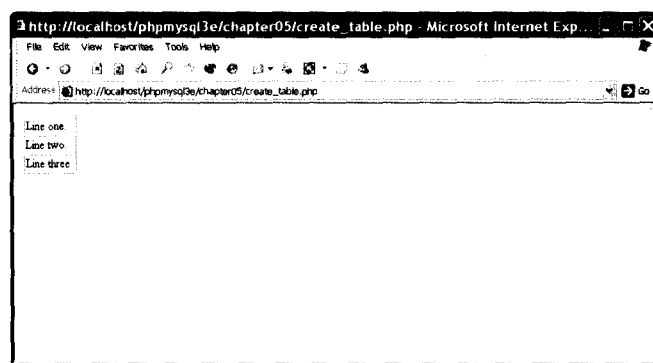



Figura 5.4 Essa tabela de HTML é o resultado de chamar create_table( ).

Passar um parâmetro permitiu obter dados que foram criados fora da função – nesse caso, o array $data – na função.

Como com funções predefinidas, as funções definidas pelo usuário podem ter diversos parâmetros e parâmetros opcionais. Podemos melhorar nossa função create_table( ) de várias maneiras, mas uma maneira poderia ser permitir ao chamador especificar a borda ou outros atributos da tabela. Eis uma versão melhorada da função. É muito semelhante, mas nos permite opcionalmente configurar a largura de borda da tabela, o espaçamento de célula e o preenchimento de célula.

```
function create_table2( $data, $border =1, $cellpadding = 4, $cellspacing = 4 )
{
  echo "<table border='$border' cellpadding = '$cellpadding'"
       ." cellspacing='$cellspacing'>";
```

```
  reset($data);
  $value = current($data);
  while ($value)
  {
     echo "<tr><td>$value</td></tr>\n";
     $value = next($data);
  }
  echo '</table>';
}
```

O primeiro parâmetro para create_table2( ) ainda é requerido. Os três próximos são opcionais porque definimos valores padrão para eles. Podemos criar saída muito semelhante a essa mostrada na Figura 5.4 com esta chamada para create_table2( ):

```
create_table2($my_array);
```

Se quisermos que os mesmos dados sejam exibidos em um estilo mais distribuído, poderíamos chamar nossa nova função da seguinte maneira:

```
create_table2($my_array, 3, 8, 8);
```

Realmente nem todos os valores opcionais precisam ser fornecidos – podemos fornecer alguns e ignorar alguns. Os parâmetros serão atribuídos da esquerda para a direita.

Lembre-se de que você não pode omitir um parâmetro opcional e incluir um parâmetro listado mais tarde. Nesse exemplo, se quiser passar um valor para cellspacing, você também terá de passar um para cellpadding. Essa é uma causa comum de erros de programação. Também é a razão por que parâmetros opcionais são especificados por último em qualquer lista de parâmetros.

A chamada de função a seguir:

```
create_table2($my_array, 3);
```

é perfeitamente válida e resultará em $border sendo configurado como 3, e $cellpadding e $cellspacing sendo configurados como padrão.

Você também pode declarar funções que aceitem um número variável de parâmetros. É possível descobrir quantos parâmetros foram passados e quais são seus valores com a ajuda de três funções: func_num_args( ), func_get_arg( ) e func_get_args( ).

Por exemplo, considere esta função:

```
function var_args( )
{
 echo "Number of parameters:";
 echo func_num_args( );

 echo '<br />';
 $args = func_get_args( );
 foreach ($args as $arg)
   echo $arg.'<br />';
}
```

Essa função reporta o número de parâmetros passados a ela e imprime cada um deles. A função func_num_args( ) retorna o número de argumentos passados. A função func_get_args( ) retorna um array de argumentos. Você também pode acessar os argumentos um de cada vez utilizando a função func_get_arg( ), passando a ela o número do argumento que deseja acessar. (Os argumentos são numerados começando do zero.)

Entendendo escopo

Você talvez tenha notado que quando precisamos utilizar variáveis dentro de um arquivo requerido ou incluído, simplesmente as declaramos no script antes da instrução require( ) ou include( ), mas

ao utilizar uma função, passamos explicitamente essas variáveis para dentro da função. Isso acontece em parte porque não existe nenhum mecanismo para passar variáveis explicitamente para um arquivo requerido ou incluído e em parte porque o escopo de variável comporta-se de maneira diferente para funções.

O escopo de uma variável controla onde essa variável é visível e utilizável. Diferentes linguagens de programação têm regras diferentes que configuram o escopo de variáveis. O PHP tem regras relativamente simples:

- Variáveis declaradas dentro de uma função estão no escopo da instrução em que são declaradas até a chave de fechamento no final da função. Isso é chamado *escopo de função*. Essas variáveis são chamadas *variáveis locais*.
- Variáveis declaradas fora de funções estão no escopo a partir da instrução em que elas são declaradas até o final do arquivo, mas *não dentro de funções*. Isso é chamado *escopo global*. Essas variáveis são chamadas *variáveis globais* .
- As variáveis especiais de superglobal são funções visíveis internas e externas. (Veja o Capítulo 1, para obter uma lista dessas variáveis.)
- Utilizar instruções `require( )` e `include( )` não afeta o escopo. Se a instrução for utilizada dentro de uma função, o escopo de função se aplica. Se não estiver dentro de uma função, o escopo global se aplica.
- A palavra-chave `global` pode ser utilizada para manualmente especificar que uma variável definida ou utilizada dentro de uma função terá escopo global.
- As variáveis podem ser manualmente excluídas chamando `unset(`*`$nome_da_variável`*`)`. Uma variável não está mais no escopo se ela foi excluída com `unset`.

Os seguintes exemplos talvez ajudem a esclarecer as dúvidas.

O código a seguir não produz nenhuma saída. Aqui estamos declarando uma variável chamada `$var` dentro da função `fn( )`. Como essa variável está declarada dentro de uma função, ela tem escopo de função e só existe a partir de onde está declarada, até o final da função. Quando novamente referenciamos `$var` fora da função, uma nova variável chamada `$var` é criada. Essa nova variável tem escopo global e será visível até o final do arquivo. Infelizmente, se a única instrução que utilizamos com essa nova variável `$var` for `echo`, ela nunca terá um valor.

```
function fn( )
{
  $var = 'contents';
}
echo $var;
```

O seguinte exemplo é o inverso. Declaramos uma variável fora da função e então tentamos utilizá-la dentro de uma função.

```
function fn( )
{
  echo 'inside the function, $var = '.$var.'<br />';
  $var = 'contents2';
  echo 'inside the function, $var = '.$var.'<br />';
}
$var = 'contents 1';
fn( );
echo 'outside the function, $var = '.$var.'<br />';
```

A saída desse código será:

```
inside the function, $var =
inside the function, $var = contents 2
outside the function, $var = contents 1
```

As funções não são executadas até que sejam chamadas, então a primeira instrução executada é `$var = 'contents 1';`. Isso cria uma variável chamada `$var`, com escopo global e o conteúdo `"contents 1"`. A próxima instrução executada é uma chamada à função `fn( )`. As linhas dentro da instrução são executadas em ordem. A primeira linha na função refere-se a uma variável chamada `$var`. Quando essa linha for executada, ela não pode ver a `$var` anterior que criamos, então ela cria uma nova com escopo de função e a ecoa. Isso cria a primeira linha de saída.

A próxima linha dentro da função configura o conteúdo de `$var` para ser `"contents 2"`. Como estamos dentro da função, essa linha altera o valor do local `$var`, não o global. A segunda linha de saída verifica se essa alteração funcionou.

A função está agora concluída, então a linha final do script é executada. Essa instrução `echo` demonstra que o valor da variável global não mudou.

Se quisermos que uma variável criada dentro de uma função seja global, podemos utilizar a palavra-chave `global` da seguinte maneira:

```
function fn( )
{
  global $var;
  $var = 'contents';
  echo 'inside the function, $var = '.$var.'<br />';
}

fn( );
echo 'outside the function, $var = '.$var.'<br />';
```

Nesse exemplo, a variável `$var` foi definida explicitamente como global, o significa que depois que a função for chamada, a variável também existirá fora da função. A saída desse script será a seguinte:

```
inside the function, $var = contents
outside the function, $var = contents
```

Observe que a variável está no escopo a partir do ponto em que a linha `global $var;` é executada. Poderíamos ter declarado a função acima ou abaixo de onde chamamos. (Note que o escopo de função é bem diferente do escopo de variável!) A localização da declaração da função não importa, o que é importante é onde chamamos a função e, portanto, onde executamos o código dentro dela.

Você também pode utilizar a palavra-chave `global` na parte superior de um script quando uma variável for utilizada pela primeira vez para declarar que ela deve estar no escopo por todo o script. Essa é possivelmente a utilização mais comum da palavra-chave `global`.

Você pode ver a partir dos exemplos precedentes que é perfeitamente válido reutilizar um nome de variável para uma variável dentro e fora de uma função sem interfirir com a outra. Entretanto, geralmente não é uma boa idéia porque sem ler cuidadosamente o código e pensar sobre escopo, as pessoas podem assumir que as variáveis são idênticas.

Passando por referência *versus* passando por valor

Se quisermos escrever uma função chamada `increment( )` que permita incrementar um valor, talvez sejamos tentados a tentar escrevê-la da seguinte maneira:

```
function increment($value, $amount = 1)
{
  $value = $value +$amount;
}
```

Esse código não terá nenhuma utilização. A saída a partir do seguinte código de teste será `"10"`.

```
$value = 10;
increment ($value);
echo $value;
```

O conteúdo de $value não mudou. Isso acontece por causa das regras de escopo. Esse código cria uma variável denominada $value que contém 10. Ela então chama a função increment( ). A variável $value na função é criada quando a função é chamada. Um é adicionado a ela, então o valor de $value é 11 dentro da função, até a função terminar e retornarmos para o código que a chamou. Nesse código, a variável $value é uma variável diferente, com escopo global e, portanto, inalterado.

Uma maneira de superar isso é declarar $value na função como global, mas isso significa que para utilizar essa função, a variável que queríamos incrementar precisaria ser denominada $value. Uma abordagem melhor seria utilizar *passagem por referência* .

A maneira normal como os parâmetros de função são chamados é denominada de *passar por valor*. Quando você passa um parâmetro, é criada uma nova variável que contém o valor passado. Ela é uma cópia da original. Você é livre para modificar esse valor de qualquer maneira, mas o valor da variável original fora da função permanece inalterado.

A melhor abordagem é utilizar *passagem por referência*. Aqui, quando um parâmetro é passado para uma função, em vez de criar uma nova variável, a função recebe uma referência para a variável original. Essa referência tem um nome de variável, começando com um sinal de cifrão e pode ser utilizada exatamente da mesma maneira que outra variável. A diferença é que em vez de ter um valor próprio, ela meramente referencia a original. Quaisquer modificações feitas na referência também afetam a original.

Especificamos que um parâmetro deve utilizar passar por referência colocando um *e* comercial (&) antes do nome de parâmetro na definição da função. Nenhuma alteração é requerida na chamada de função.

O exemplo increment( ) precedente pode ser modificado para ter um parâmetro passado por referência, e funcionará corretamente.

```
function increment(&$value, $amount = 1)
{
  $value = $value +$amount;
}
```

Agora temos uma função operacional e somos livres para nomear a variável que queremos incrementar da maneira como quisermos. Como já foi mencionado, é confuso para os humanos utilizar o mesmo nome dentro e fora de uma função, então daremos um novo nome para a variável no script principal. O seguinte código de teste agora ecoará 10 antes da chamada a increment( ) e 11 depois.

```
$a = 10;
echo $a.'<br />';
increment ($a);
echo $a.'<br />';
```

Retornando a partir de funções

A palavra-chave return pára a execução de uma função. Quando uma função termina porque todas instruções foram executadas ou a palavra-chave return é utilizada, a execução retorna à instrução depois da chamada de função.

Se você chamar a seguinte função, apenas a primeira instrução echo será executada.

```
function test_return( )
{
  echo 'This statement will be executed';
  return;
  echo 'This statement will never be executed';
}
```

Obviamente, isso não é uma maneira muito útil de empregar return. Normalmente, você só vai querer retornar a partir do meio de uma função em resposta ao fato de uma condição ser atendida.

Uma condição de erro é uma razão comum de utilizar uma instrução return para interromper a execução de uma função antes do seu final. Se, por exemplo, você escreveu uma função para descobrir qual dos dois números era maior, talvez queira sair se algum dos números estiver faltando.

```
function larger( $x, $y )
{
  if (!isset($x)||!isset($y))
  {
    echo 'This function requires two numbers';
    return;
  }
  if ($x>=$y)
    echo $x;
  else
    echo $y;
  echo '<br />';
}
```

A função predefinida isset( ) informa se uma variável foi criada e recebeu um valor. Nesse código, vamos fornecer uma mensagem de erro e retornar se algum dos parâmetros não foi configurado com um valor. Testamos isso utilizando !isset( ), significando "NOT isset( )", então a instrução if pode ser lida como "se x não estiver configurado ou se y não estiver configurado". A função retornará se qualquer uma dessas condições for verdadeira.

Se a instrução return for executada, as linhas subseqüentes de código na função serão ignoradas. A execução de programa retornará para o ponto em que a função foi chamada. Se ambos os parâmetros estiverem configurados, a função ecoará o maior dos dois parâmetros.

A saída a partir do código a seguir,

```
$a = 1;
$b = 2.5;
$c = 1.9;
larger($a, $b);
larger($c, $a);
larger($d, $a);
```

será da seguinte maneira:

```
2.5
1.9
This function requires two numbers
```

Retornando valores de funções

Sair de uma função não é a única razão de utilizar return. Muitas funções utilizam instruções return para se comunicar com o código que as chamaram. Em vez de ecoar o resultado da comparação na função larger( ), nossa função poderia ser mais útil se retornássemos a resposta. Dessa maneira, o código que chamou a função pode escolher se e como exibir ou utilizá-la. A função predefinida equivalente max( ) comporta-se dessa maneira.

Podemos escrever a função larger( ) da seguinte maneira:

```
function larger ($x, $y)
{
  if (!isset($x)||!isset($y))
    return false;
  else if ($x>=$y)
    return $x;
  else
    return $y;
}
```

Aqui estamos retornando o maior dos dois valores passados. Obviamente retornaremos um valor diferente no caso de um erro. Se um dos números estiver faltando, retornaremos `false`. A única armadilha dessa abordagem é que os programadores que chamam a função devem testar o tipo de retorno com `===` para se certificar de que `false` não seja confundido com 0.

Para fins de comparação, a função predefinida `max( )` não retorna nada se ambas as variáveis não forem configuradas; e se apenas uma tiver sido configurada, a função retorna esta.

O código a seguir,

```
$a = 1; $b = 2.5; $c = 1.9;
echo larger($a, $b)."<br />";
echo larger($c, $a)."<br />";
echo larger($d, $a)."<br />";
```

produzirá esta saída porque `$d` não existe e `false` não é visível:

```
2.5
1.9
```

As funções que realizam alguma tarefa, mas não precisam retornar um valor, retornam freqüentemente `true` ou `false` para indicar se foram bem-sucedidas ou se falharam. Os valores booleanos `true` e `false` podem ser representados com valores inteiros 1 e 0 respectivamente, embora sejam de tipos diferentes.

Blocos de código

Declaramos que um grupo de instruções é um bloco colocando-as entre chaves. Isso não afeta a maior parte da operação de seu código, mas tem implicações específicas incluindo a maneira como as estruturas de controle como loops e condicionais executam.

Os dois exemplos seguintes funcionam de modo muito diferente:

Exemplo sem bloco de código

```
for($i = 0; $i < 3; $i++ )
  echo 'Line 1<br />';
echo 'Line 2<br />';
```

Exemplo com bloco de código

```
for($i = 0; $i < 3; $i++ )
{
  echo 'Line 1<br />';
  echo 'Line 2<br />';
}
```

Nos dois exemplos, o loop `for` é iterado por três vezes. No primeiro exemplo, somente a única linha logo abaixo dessa é executada pelo loop `for`. A saída desse exemplo é da seguinte maneira:

```
Line 1
Line 1
Line 1
Line 2
```

O segundo exemplo utiliza um bloco de código para agrupar duas linhas juntas. Isso significa que as duas linhas são executadas três vezes pelo loop `for`. A saída desse exemplo é:

```
Line 1
Line 2
Line 1
```

```
Line 2
Line 1
Line 2
```

Como o código nesses exemplos é recuado adequadamente, você provavelmente pode ver a diferença entre eles apenas com uma visão geral. O recuo do código é destinado a oferecer uma interpretação visual das linhas que são afetadas pelo loop `for`. Mas note que os espaços não afetam como o PHP processa o código.

Em algumas linguagens, os blocos de código afetam o escopo variável. Esse não é o caso no PHP.

Implementando recursão

Funções recursivas são suportadas no PHP. Uma *função recursiva* é uma função que chama a si própria. Essas funções são particularmente úteis para navegar por estruturas de dados dinâmicas como listas vinculadas e árvores.

Entretanto, poucas aplicações baseadas na Web requerem uma estrutura de dados com essa complexidade, de modo que a utilização de recursão é mínima. A recursão pode ser utilizada no lugar da iteração em muitos casos porque ambas permitem fazer algo repetidamente. As funções recursivas são mais lentas e utilizam mais memória que a iteração, então você deve utilizar iteração onde for possível.

Para sermos completos, veremos um exemplo breve mostrado na Listagem 5.5.

Listagem 5.5 **`recursion.php` – É simples inverter uma string utilizando recursão – a versão iterativa também é mostrada**

```
function reverse_r($str)
{
   if (strlen($str)>0)
     reverse_r(substr($str, 1));
   echo substr($str, 0, 1);
   return;
}

function reverse_i($str)
{
  for ($i=1; $i<=strlen($str); $i++)
   {
     echo substr($str, -$i, 1);
   }
   return;
}
```

Nessa listagem, implementamos duas funções. Ambas imprimirão uma string invertida. A função `reverse_r( )` é recursiva e `reverse_i( )` é iterativa.

A função `reverse_r( )` aceita uma string como parâmetro. Quando você for chamá-la, ela continuará a chamar a si própria, passando a cada vez desde o segundo até o último caractere da string. Por exemplo, se você chamar

```
reverse_r('Hello');
```

ela chamará a si própria várias vezes, com os parâmetros a seguir:

```
reverse_r('ello');
reverse_r('llo');
reverse_r('lo');
reverse_r('o');
reverse_r('');
```

Cada chamada que a função faz a si própria faz uma nova cópia do código de função na memória do servidor, mas com um parâmetro diferente. É como fingir que realmente estamos chamando uma função diferente toda vez. Isso evita a confusão entre instâncias da mesma função.

A cada chamada, o comprimento da string passada é testado. Quando alcançamos o final da string (`strlen( )==0`), a condição falha. A instância mais recente da função (`reverse_r('')`) então continuará e realizará a próxima linha de código, o que deve ecoar o primeiro caractere da string que lhe foi passada – nesse caso, não há nenhum caractere porque a string está vazia.

Em seguida, essa instância da função retorna controle à instância que a chamou, isto é `reverse_r('o')`. Esta imprime o primeiro caractere em sua string – "o" – e retorna controle à instância que a chamou.

O processo continua – imprimindo um caractere e então retornando para a instância da função acima dela na ordem de chamadas – até que o controle seja retornado outra vez para o programa principal.

Há algo muito elegante e matemático sobre soluções recursivas. Mas, na maioria dos casos, é melhor utilizar uma solução iterativa. O código para isso também está na Listagem 5.5. Observe que não é mais longa e faz exatamente o mesmo (embora isso não aconteça sempre com funções iterativas). A diferença principal é que a função recursiva fará cópias de si própria na memória e incorrerá no overhead de diversas chamadas de função.

Talvez você escolha utilizar uma solução recursiva quando o código for muito mais curto e mais elegante que a versão iterativa, mas isso não acontecerá freqüentemente no domínio dessa aplicação.

Embora a recursão pareça mais elegante, os programadores se esquecem freqüentemente de fornecer uma condição de terminação para a recursão. Isso significa que função a recorrerá até que o servidor fique sem memória ou até que o tempo de execução máximo seja excedido, o que ocorrer primeiro.

Leitura adicional

O uso de `include( )`, `require( )`, `function` e `return` também são explicados no manual on-line. Para descobrir mais sobre conceitos como recursão, passar por valor/referência e escopo que afeta muitas linguagens, você pode ver um manual de computação geral, como C++ *How To Program* de Dietel & Dietel.

A seguir

Agora que você está utilizando arquivos include, arquivos require e funções para tornar seu código mais fácil de manter e reutilizável, o próximo capítulo aborda software orientado a objetos e o suporte oferecido no PHP. Utilizar objetos permite alcançar objetivos semelhantes aos conceitos apresentados neste capítulo, mas ainda com maiores vantagens para projetos complexos.

6

PHP orientado a objetos

ESTE CAPÍTULO EXPLICA CONCEITOS DE DESENVOLVIMENTO orientado a objetos e mostra como podem ser implementados no PHP.

Os tópicos-chave neste capítulo incluem:

- Conceitos de orientação a objetos;
- Classes, atributos e operações;
- Atributos de classes;
- Constantes de classe;
- Chamada a métodos de classe;
- Herança;
- Modificadores de acesso;
- Métodos estáticos;
- Type hinting;
- Clonagem de objetos;
- Classes abstratas;
- Projeto de classes;
- Implementação do projeto;
- Funcionalidade avançada de orientação a objetos (OO);
- Nova funcionalidade de orientação a objetos (OO) do PHP5.

Conceitos de orientação a objetos

As linguagens de programação modernas normalmente suportam ou até mesmo exigem uma abordagem orientada a objetos para desenvolvimento de software. O desenvolvimento orientado a objetos (Object-Oriented – OO) tenta utilizar as classificações, os relacionamentos e as propriedades dos objetos no sistema para ajudar no desenvolvimento de programa.

Classes e objetos

No contexto de software OO, um objeto pode ser quase qualquer item ou conceito – um objeto físico como uma escrivaninha ou um cliente; ou um objeto conceitual que só exista no software, como

uma área de entrada de texto ou um arquivo. Geralmente, estamos mais interessados em objetos conceituais incluindo objetos do mundo real que precisam ser representados no software.

O software orientado a objetos é projetado e construído como um conjunto de objetos autocontidos tanto com atributos como com operações que interagem para atender às nossas necessidades. Os *atributos* são propriedades ou variáveis que se relacionam ao objeto. As *operações* são métodos, ações ou funções que o objeto pode realizar para modificar a si próprio ou para algum efeito externo.

A principal vantagem do software orientado a objetos é sua capacidade de suportar e incentivar o uso do *encapsulamento* – também conhecido como ocultamento de dados. Essencialmente, o acesso aos dados dentro de um objeto é sempre disponível via operações do objeto, conhecidas como *interface* do objeto.

A funcionalidade de um objeto está limitada aos dados que ele utiliza. Podemos facilmente alterar os detalhes de como o objeto é implementado para melhorar o desempenho, adicionar novos recursos ou corrigir bugs *sem alterar a interface*, o que pode ter efeitos por todo o projeto.

Em outras áreas de desenvolvimento de software, OO é a norma e software orientado à função é considerado fora de moda. Por várias razões, a maioria dos scripts Web infelizmente ainda é projetada e escrita utilizando uma abordagem *ad hoc* seguindo uma metodologia orientada à função.

Existem várias razões para isso. A maioria dos projetos Web é relativamente pequena e simples. Você pode escapar das conseqüências de pegar um serrote e construir uma prateleira de madeira para temperos sem planejar sua abordagem e pode completar com sucesso a maior parte dos projetos de software Web da mesma maneira por causa de seu pequeno tamanho. Entretanto, se pegou um serrote e tentou construir uma casa sem planejamento formal, você não obtém resultados de qualidade, se é que conseguirá obter algum resultado – o mesmo é verdadeiro para grandes projetos de software.

Muitos projetos Web desenvolvem-se a partir de um conjunto de páginas com hiperlink para uma aplicação complexa. Essas aplicações complexas, quer sejam apresentadas via caixas de diálogo e janelas ou via páginas HTML dinamicamente geradas, precisam de uma metodologia de desenvolvimento adequadamente elaborada. A orientação a objeto pode ajudar a gerenciar a complexidade nos projetos, aumentar capacidade de reutilização de código e assim reduzir custos de manutenção.

No software OO, um objeto é uma coleção identificável e única de dados armazenados e operações que operam nesses dados. Por exemplo, poderíamos ter dois objetos que representam botões. Mesmo se ambos tiverem um rótulo "OK", uma largura de 60 pixels, uma altura de 20 pixels e qualquer outro atributo que seja idêntico, ainda assim precisamos ser capazes de lidar com um botão ou o outro. Em software, temos variáveis separadas que atuam como *handles* (identificadores únicos) para os objetos.

Os objetos podem ser agrupados em classes. As classes representam um conjunto de objetos que pode variar de indivíduo para indivíduo, mas deve ter uma certa quantidade em comum. Uma classe contém objetos em que todos têm as mesmas operações comportando-se da mesma maneira e os mesmos atributos representando as mesmas coisas, embora os valores desses atributos variem de objeto para objeto.

O substantivo bicicleta pode ser pensado como uma classe de objetos que descreve muitas bicicletas distintas com muitos recursos, ou *atributos*, em comum – como duas rodas, uma cor e um tamanho e operações, como movimento.

Minha bicicleta pode ser imaginada como um objeto que se ajusta na classe bicicleta. Ela tem todos os recursos em comum de todas as bicicletas incluindo uma operação de movimento que se comporta da mesma maneira como o movimento da maioria das outras bicicletas – mesmo se for raramente utilizada. Os atributos da minha bicicleta têm valores únicos porque minha bicicleta é verde e nem todas as bicicletas são dessa cor.

Polimorfismo

Uma linguagem de programação orientada a objetos deve suportar *polimorfismo*, o que significa que classes diferentes podem ter comportamentos diferentes para a mesma operação. Se, por exem-

plo, tivermos uma classe carro e uma classe bicicleta, as duas classes podem ter diferentes operações de movimento. Para objetos do mundo real, isso raramente seria um problema. As bicicletas provavelmente não ficariam confusas utilizando em vez disso a operação de movimento de um carro. Entretanto, uma linguagem de programação não possui o senso comum do mundo real, então a linguagem deve suportar polimorfismo para saber qual operação de movimento utilizar em um objeto particular.

O polimorfismo é mais uma característica dos comportamentos que dos objetos. No PHP, somente funções de membro de uma classe podem ser polimórficas. Uma comparação do mundo real é a de verbos nos idiomas naturais, que são equivalentes a funções de membro. Considere as maneiras como uma bicicleta pode ser utilizada na vida real. Você pode limpá-la, movê-la, desmontá-la, consertá-la ou pintá-la, entre outras coisas.

Esses verbos descrevem ações genéricas porque você não conhece o tipo de objeto sobre o qual essas ações atuam. (Esse tipo de abstração de objetos e ações é uma das características distintivas da inteligência humana.)

Por exemplo, mover uma bicicleta requer ações completamente diferentes daquelas requeridas para mover um carro, mesmo que os conceitos sejam semelhantes. O verbo *mover* pode ser associado a um conjunto particular de ações apenas quando o objeto sobre o qual elas atuam torna-se conhecido.

Herança

A *herança* permite criar um relacionamento hierárquico entre classes utilizando *subclasses*. Uma subclasse herda atributos e operações de sua *superclasse*. Por exemplo, carro e bicicleta têm algumas coisas em comum. Poderíamos utilizar uma classe veículo para conter características como um atributo de cor e uma operação de movimento que todos os veículos têm e então deixar que as classes carro e bicicleta herdem de veículo.

Com herança, você pode construir e adicionar a classes existentes. A partir de uma classe básica simples, você pode derivar classes especializadas e mais complexas conforme necessário. Isso torna o código mais reutilizável, o que é uma das vantagens importantes de uma abordagem orientada a objetos.

Utilizar herança pode poupar trabalho se operações puderem ser escritas uma vez em uma superclasse em vez de muitas vezes em subclasses separadas. Isso talvez permita modelar mais exatamente os relacionamentos do mundo real. Se uma frase sobre duas classes fizer sentido com "é uma" entre as classes, a herança é provavelmente apropriada. A frase "um carro é um veículo" faz sentido, mas a frase "um veículo é um carro" não faz sentido porque nem todos os veículos são carros. Portanto, carro pode herdar a partir de veículo.

Criando classes, atributos, operações no PHP

Até agora, discutimos classes de maneira relativamente abstrata. Ao criar uma classe no PHP, você deve utilizar a palavra-chave `class`.

Estrutura de uma classe

Uma definição mínima de classe é parecida com o seguinte:

```
class classname
{
}
```

Para serem úteis, as classes precisam de atributos e operações. Criamos atributos declarando variáveis dentro de uma definição de classe utilizando a palavra-chave `var`. O seguinte código cria uma classe chamada `classname` com dois atributos, `$attribute1` e `$attribute2`.

```
class classname
{
  var $attribute1;
  var $attribute2;
}
```

Criamos operações declarando funções dentro da definição de classe. O código a seguir criará uma classe chamada `classname` com duas operações que não fazem nada. A operação `operation1( )` não aceita nenhum parâmetro e `operation2( )` aceita dois parâmetros.

```
class classname
{
  function operation1( )
  {
  }
  function operation2($param1, $param2)
  {
  }
}
```

Construtores

A maioria das classes terá um tipo de operação especial denominado *construtor*. Um *construtor* é chamado quando um objeto é criado e ele normalmente também realiza tarefas úteis de inicialização como configurar atributos como valores iniciais sensíveis ou criar outros objetos necessários para esse objeto.

Um construtor é declarado da mesma maneira que outras operações, mas tem o mesmo nome que a classe. Embora possamos chamar o construtor manualmente, seu propósito principal é ser chamado automaticamente quando um objeto é criado. O seguinte código declara uma classe com um construtor:

```
class classname
{
  function classname($param)
  {
    echo "Constructor called with parameter $param <br />";
  }
}
```

O PHP5 agora oferece suporte a sobrecarga de funções, o que significa que você pode fornecer mais de uma função com o mesmo nome e diferentes números ou tipos de parâmetros. (Esse recurso tem suporte em muitas linguagens orientadas a objetos.) Discutiremos isso posteriormente neste capítulo.

Destruidores

O oposto do construtor é o *destruidor*. Esse é um novo recurso no PHP5. Ele permite que você tenha alguma funcionalidade que será executada logo antes de uma classe ser destruída, o que ocorrerá automaticamente quando todas as referências a uma classe tiverem sido desconfiguradas ou estiverem fora de escopo.

Assim como o construtor, o destruidor de uma classe deve ser nomeado `__destruct( )`. Destruidores não aceitam parâmetros.

Instanciação de classes

Depois de declararmos uma classe, precisamos criar um objeto – um indivíduo particular que seja membro da classe – com que trabalhar. Isso também é conhecido como criar uma instância ou ins-

tanciar uma classe. Criamos um objeto utilizando a palavra-chave new. Precisamos especificar a classe da qual o objeto será uma instância e fornecer qualquer parâmetro requerido por nosso construtor.

O seguinte código declara uma classe denominada classname com um construtor e então cria três objetos do tipo classname:

```
class classname
{
  function classname($param)
  {
    echo "Constructor called with parameter $param <br />";
  }
}

$a = new classname('First');
$b = new classname('Second');
$c = new classname( );
```

Como o construtor é chamado toda vez que criamos um objeto, esse código produz a saída a seguir:

```
Constructor called with parameter First
Constructor called with parameter Second
Constructor called with parameter
```

Utilizando atributos de classe

Dentro de uma classe, você tem acesso a uma variável especial chamada $this. Se um atributo de sua classe atual se chama $attribute, você o referencia como $this->attribute quando configurar ou acessar a variável a partir de uma operação dentro da classe.

O seguinte código demonstra como configurar e acessar um atributo dentro de uma classe:

```
class classname
{
  var $attribute;
  function operation($param)
  {
    $this->attribute = $param
    echo $this->attribute;
  }
}
```

Geralmente, não é uma boa idéia acessar diretamente os atributos de fora de uma classe. Uma das vantagens da abordagem orientada a objetos é que ela encoraja o encapsulamento. Você pode forçá-lo utilizando as funções __get e __set. Se, em vez de acessar os atributos de uma classe diretamente, você escrever funções acessoras, pode fazer todos os acessos com uma única seção de código. Ao escrever as funções acessoras, elas devem se parecer com o seguinte:

```
class classname
{
 var $attribute;
 function __get($name)
 {
   return $this->$name;
 }
 function __set ($name, $value)
 {
   $this->$name = $value;
 }
}
```

Esse código simplesmente fornece funções mínimas para acessar o atributo chamado $attribute. Temos uma função chamada get_attribute( ) que simplesmente retorna o valor de $attribute e uma função chamada set_attribute( ) que atribui um novo valor a $attribute.

Observe que __get( ) utiliza um parâmetro – o nome de um atributo – e retorna o valor do atributo. Da mesma forma, a função __set( ) utiliza dois parâmetros: o nome de um atributo e o valor que você deseja definir para ele.

Você não chama essas funções diretamente. O sublinhado duplo antes do nome mostra que essas funções possuem um significado especial no PHP, assim como as funções __construct( ) e __destruct( ).

Então, como elas funcionam? Se você instanciar a classe

```
$a = new classname( );
```

pode então utilizar as funções __get( ) e __set( ) para verificar e definir o valor de qualquer atributo.

Se digitar

```
$a->$attribute = 5;
```

essa instrução chama a função __set( ) com o valor de $name definido em "attribute", e o valor de $value é definido para 5. É necessário escrever a função __set( ) para realizar qualquer verificação de erro desejada.

A função __get( ) funciona de maneira semelhante. Se no código, você referencia

```
$a->attribute
```

essa expressão implicitamente chama a função __get( ) com o parâmetro $name definido para "attribute". Cabe a você escrever a função __get( ) para retornar o valor.

À primeira vista, esse código talvez pareça adicionar pouco ou nenhum valor. Em sua forma atual, isso provavelmente é verdadeiro, mas a razão de fornecer funções acessoras é simples: teremos então somente uma seção de código que acessa esse atributo particular.

Com apenas um único ponto de acesso, você pode implementar verificações para garantir que apenas os dados importantes sejam armazenados. Se, mais tarde, você considerar que o valor de $attribute deve estar entre 0 e 100, poderá adicionar algumas linhas de código uma vez e verificar antes de aceitar as mudanças. Você pode mudar a função __set( ) da seguinte maneira:

```
function __set ($name, $value)
{
 if( $name='attribute' && $value >= 0 && $value <= 100 )
   $this->attribute = $value;
}
```

Com apenas um único ponto de acesso, estamos livres para alterar a implementação subjacente. Se por alguma razão, escolhemos alterar a maneira como $attribute é armazenada, as funções acessoras permitem fazer isso e somente alterar o código em um lugar.

Poderíamos decidir que em vez de armazenar $attribute como uma variável, apenas iremos recuperá-la de um banco de dados quando necessário, calcularemos um valor atualizado toda vez que ele for solicitado, inferiremos um valor a partir de valores de outros atributos ou codificaremos nossos dados como um tipo de dados menor. Independente do que decidamos fazer, podemos simplesmente modificar nossas funções acessoras. Outras seções de código não serão afetadas contanto que façamos as funções acessoras ainda aceitarem ou retornarem os dados que outras partes do programa esperam.

Controlando acesso com **private** e **public**

O PHP5 introduziu os modificadores de acesso, que controlam a visibilidade de atributos e métodos, e são inseridos na frente de declarações de atributos e métodos. O PHP5 oferece suporte a três diferentes modificadores de acesso:

- A opção default é public, o que significa que se você especificar um modificador de acesso para um atributo ou método, ele será public. Os itens que são públicos podem ser acessados de dentro ou fora da classe.
- O modificador de acesso private significa que o item marcado pode ser acessado apenas de dentro da classe. Você pode utilizá-lo em todos os atributos se não estiver usando __get( ) e __set( ). Também pode escolher tornar alguns métodos privados, por exemplo, se forem funções de utilidade para uso apenas dentro da classe. Os itens privados não serão herdados (veremos mais sobre esse assunto posteriormente neste capítulo).
- O modificador de acesso protected significa que o item marcado pode ser acessado apenas de dentro da classe. Ele também existe em qualquer subclasse; mais uma vez, voltaremos a esse assunto quando discutirmos herança, posteriormente neste capítulo. Por enquanto, você pode pensar em protected como sendo o meio-termo entre private e public.

Para acrescentar modificadores de acesso à classe de exemplo, altere o código da seguinte maneira:

```
class classname
{
 public $attribute;
 public function __get($name)
 {
   return $this->$name;
 }
 public function __set ($name, $value)
 {
   $this->$name = $value;
 }
}
```

Aqui, cada membro da classe é precedido por um modificador de acesso, para mostrar se ele é privado ou público. Você pode deixar a palavra-chave public de fora porque ela é o padrão, mas, com ela, o código fica mais fácil de entender se os outros modificadores estão sendo utilizados.

Observe que removemos a palavra-chave var da frente do atributo; ela foi substituída pelo modificador de acesso public. Nesse caso, tornamos tudo público.

Chamando operações de classe

Podemos chamar operações de maneira muito semelhante da que chamamos atributos de classe. Se tivermos a classe

```
class classname
{
  function operation1( )
  {
  }
  function operation2($param1, $param2)
  {
  }
}
```

e criarmos um objeto de tipo classname chamado $a da seguinte maneira:

```
$a = new classname( );
```

Então chamamos operações da mesma maneira como chamamos outras funções: utilizando seu nome e colocando todos os parâmetros que elas precisam entre colchetes. Como essas operações pertencem a um objeto em vez de funções normais, precisamos especificar o objeto a qual elas pertencem. O nome de objeto é utilizado da mesma maneira como os atributos de um objeto:

```
$a->operation1( );
$a->operation2(12, 'test');
```

Se nossas operações retornam algo, podemos capturar esses dados de retorno da seguinte maneira:

```
$x = $a->operation1( );
$y = $a->operation2(12, 'test');
```

Implementando herança no PHP

Se nossa classe deve ser uma subclasse de outra, você pode utilizar a palavra-chave `extends` para especificar isso. O próximo código cria uma classe chamada B que herda a partir de alguma classe anteriormente definida chamada A.

```
class B extends A
{
  var $attribute2;
  function operation2( )
  {
  }
}
```

Se a classe A foi declarada

```
class A
{
  var $attribute1;
  function operation1( )
  {
  }
}
```

todos os seguintes acessos a operações e atributos de um objeto de tipo B seriam válidos:

```
$b = new B( );
$b->operation1( );
$b->attribute1 = 10;
$b->operation2( );
$b->attribute2 = 10;
```

Observe que como a classe B estende a classe A, você pode referenciar `operation1( )` e `$attribute1`, embora esses tenham sido declarados na classe A. Como uma subclasse de A, B tem a mesma funcionalidade e dados. Além disso, B declarou um atributo e uma operação próprios.

É importante notar que herança só funciona em uma direção. A subclasse ou filho herda recursos de seu pai ou superclasse, mas o pai não adquire recursos do filho. Isso significa que as duas últimas linhas neste código estão erradas:

```
$a = new A( );
$a->operation1( );
$a->attribute1 = 10;
$a->operation2( );
$a->attribute2 = 10;
```

A classe A não tem uma `operation2( )` ou uma `attribute2`.

Controlando visibilidade através de herança com **private** e **protected**

Você pode utilizar os modificadores de acesso `private` e `protected` para controlar o que é herdado. Se um atributo ou método for especificado como `private`, ele não será herdado. Se um atributo ou

método for especificado como protected, não será visível de fora da classe (como um elemento private), mas será herdado.

Considere o seguinte exemplo:

```
<?php
class A
{
 private function operation1( )
 {
   echo "operation1 called";
 }
 protected function operation2( )
 {
   echo "operation2 called";
 }
 public function operation3( )
 {
   echo "operation3 called";
 }
}

class B extends A
{
 function __construct( )
 {
   $this->operation1( );
   $this->operation2( );
   $this->operation3( );
 }
}

$b = new B;

?>
```

Esse código cria uma operação de cada tipo na classe A: public, protected e private. B herda de A. No construtor de B, você tenta chamar as operações a partir do pai.

A linha

```
$this->operation1( );
```

produz um erro fatal como o seguinte:

```
Fatal error: Call to private method A::operation1( ) from context 'B'
```

Esse exemplo mostra que as operações privadas não podem ser chamadas a partir de uma classe filha.

Se você retirar essa linha, as duas outras chamadas de função terão êxito. A função protected é herdada, mas pode ser usada apenas a partir de dentro da classe filha, como feito aqui. Se você tentar acrescentar a linha

```
$b->operation2( );
```

ao final do arquivo, receberá o seguinte erro:

```
Fatal error: Call to protected method A::operation2( ) from context ''
```

Entretanto, você pode chamar operation3( ) de fora da classe, desta forma:

```
$b->operation3( );
```

É possível fazer essa chamada porque ela é declarada como public.

Sobrescrevendo

Neste capítulo, mostramos uma subclasse declarando novos atributos e operações. Também é válido, e às vezes útil, redeclarar os mesmos atributos e operações. Poderíamos fazer isso para fornecer a um atributo na subclasse um valor padrão diferente do mesmo atributo em sua superclasse, ou para fornecer a uma operação na subclasse uma funcionalidade diferente da mesma operação em sua superclasse. Isso é chamado de *sobrescrever (overriding)*. Por exemplo, se tivermos uma classe A:

```
class A
{
  var $attribute = 'default value';
  function operation( )
  {
    echo 'Something<br />';
    echo "The value of \$attribute is $this->attribute<br />";
  }
}
```

e quisermos alterar o valor padrão de `$attribute` e fornecer nova funcionalidade à `operation( )`, podemos criar a classe B a seguir, que sobrescreve `$attribute` e `operation( )`:

```
class B extends A
{
  var $attribute = 'different value';
  function operation( )
  {
    echo 'Something else<br />';
    echo "The value of \$attribute is $this->attribute<br />";
  }
}
```

Declarar B não afeta a definição original de A. Considere as duas linhas de código a seguir:

```
$a = new A( );
$a -> operation( );
```

Criamos um objeto do tipo A e chamamos sua função `operation( )`. Isso produzirá

```
Something
The value of $attribute is default value
```

provando que criar B não alterou A. Se criarmos um objeto do tipo B, obteremos uma saída diferente.

O código

```
$b = new B( );
$b -> operation( );
```

produzirá

```
Something else
The value of $attribute is different value
```

Da mesma maneira que fornecer novos atributos ou operações em uma subclasse não afeta a superclasse, sobrescrever atributos ou operações em uma subclasse não afeta a superclasse.

Uma subclasse herdará todos os atributos e operações de sua superclasse, a menos que você forneça substituições. Se você fornecer uma definição substituta, esta aceita precedência e sobrescreve a definição original.

A palavra-chave `parent` permite chamar a versão original da operação na classe pai. Por exemplo, para chamar `A::operation` de dentro da classe `B`, você pode usar:

```
parent::operation( );
```

No entanto, a saída produzida é diferente. Embora você possa chamar a operação a partir da classe pai, o PHP utiliza os valores de atributo da classe atual. Portanto, obtém-se a seguinte saída:

```
Something
The value of $attribute is different value
```

A herança pode ter muitas camadas de profundidade. Podemos declarar uma classe chamada C que estende B e, portanto, herda os recursos de B e de A do pai de B. A classe C pode novamente escolher quais atributos e operações dos seus pais sobrescrever e substituir.

Evitando a herança e sobrescrevendo com `final`

O PHP5 apresenta uma nova palavra-chave: `final`. Ao utilizá-la na frente de uma declaração de função, a função não pode ser sobrescrita em nenhuma subclasse. Por exemplo, você pode acrescentar isto à classe `A` no exemplo anterior, assim:

```
class A
{
 var $attribute = 'default value';
 final function operation( )
 {
   echo 'Something<br />';
   echo "The value of \$attribute is $this->attribute<br />";
 }
}
```

Essa abordagem evita que você sobrescreva `operation( )` na classe `B`. Se você tentar fazê-lo, receberá o seguinte erro:

```
Fatal error: Cannot override final method A::operation( )
```

A palavra-chave `final` também pode ser utilizada para evitar que uma classe tenha subclasses. Para evitar que a classe `A` tenha subclasses, acrescente-a desta maneira:

```
final class A
{...}
```

Se depois tentar herdar a partir de A, você receberá um erro semelhante a:

```
Fatal error: Class B may not inherit from final class (A)
```

Entendendo a herança múltipla

Algumas linguagens OO suportam múltipla herança, mas o PHP não suporta. Isso significa que cada classe só pode herdar de um pai. Não existe nenhuma restrição a quantos filhos um único pai pode compartilhar. Talvez o que isso significa não pareça imediatamente claro. A Figura 6.1 mostra três maneiras diferentes como as três classes chamadas A, B e C podem herdar.

A combinação da esquerda mostra a classe C que herda da classe B que, por sua vez, herda da classe A. Cada classe tem no máximo um pai; portanto, isso é uma herança simples perfeitamente válida no PHP.

A combinação do centro mostra as classes B e C que herdam da classe A. Cada classe tem no máximo um pai; portanto, novamente isso é uma herança única válida.

A combinação da direita mostra a classe C herdando tanto da classe A como da B. Nesse caso, a classe C tem dois pais; portanto, isso é uma herança múltipla e é inválida no PHP.

Implementando Interfaces

Interfaces são outra novidade no PHP5. Elas são vistas como alternativas para herança múltipla e são semelhantes à implementação de interfaces suportada por outras linguagens orientadas a objetos, incluindo Java.

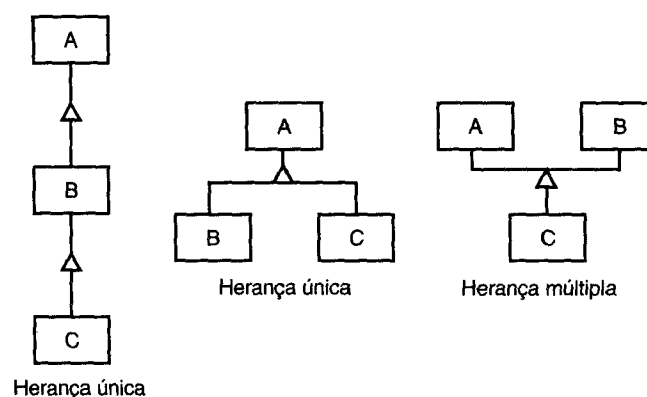



Figura 6.1 O PHP não suporta herança múltipla.

A idéia é que a interface especifica um conjunto de funções que devem ser implementadas em classes que implementam aquela interface. Por exemplo, você poderia decidir que tem um conjunto de classes que precisam ser capazes de exibir a si mesmas. Em vez de ter uma classe pai com uma função `display( )` da qual todas herdam e que a sobrescrevem, você pode implementar uma interface da seguinte forma:

```
interface Displayable
{
 function display( );
}

class webPage implements Displayable
{
 function display( )
 {
  // ...
 }
}
```

Esse exemplo ilustra um tipo de herança múltipla em que a classe `webPage` pode herdar de uma classe e implementar uma ou mais interfaces.

Se você não implementar os métodos especificados na interface (nesse caso, `display( )`), receberá um erro fatal.

Projetando classes

Agora que sabemos alguns conceitos por trás de objetos e classes e a sintaxe para implementá-los no PHP, é hora de ver como projetar classes úteis.

Muitas classes no seu código representarão classes ou categorias de objetos do mundo real. As classes que você talvez utilize no desenvolvimento Web podem incluir páginas, componentes de interface com o usuário, carrinho de compras, tratamento de erro, categorias de produto ou clientes.

Os objetos no código também podem representar instâncias específicas das classes anteriormente mencionadas, por exemplo, a home page, um botão particular ou o carrinho de compras em utilização por Fred Smith em um determinado momento. O próprio Fred Smith pode ser representado por um objeto do tipo cliente. Cada item que Fred compra pode ser representado como um objeto, pertencendo a uma categoria ou classe.

No capítulo anterior, utilizamos arquivos include simples para fornecer à empresa fictícia, a TLA Consulting, uma aparência e um comportamento consistentes por meio das diferentes páginas do seu Web site. Utilizando classes e a capacidade de economia de tempo da herança, podemos criar uma versão mais avançada do mesmo site.

Queremos ser capazes de rapidamente criar páginas para a TLA que tenham a mesma aparência e comportamento. Essas páginas devem ser capazes de ser modificadas para se ajustarem às diferentes partes do site.

Criaremos uma classe Page para esse exemplo. O objetivo principal dessa classe é limitar a quantidade de HTML necessária para criar uma nova página. Essa classe deve permitir alterar as partes que mudam de página para página, e automaticamente gerar os elementos que permanecem os mesmos. A classe deve fornecer uma estrutura flexível para criar novas páginas e não deve comprometer nossa liberdade.

Como estamos gerando nossa página a partir de um script em vez de com HTML estática, podemos adicionar quantas coisas inteligentes quisermos incluindo funcionalidade para permitir o seguinte:

- Somente alterar elementos de página em um lugar. Se alterarmos o aviso de direitos autorais ou adicionarmos um botão extra, só devemos precisar fazer a alteração em um único lugar.
- Ter conteúdo padrão para a maioria das partes da página, mas ser capaz de modificar cada elemento onde for requerido, configurando valores personalizados para elementos como o título e as metatags.
- Reconhecer que página está sendo visualizada e alterar os elementos de navegação de maneira correspondente – não há nenhum problema em ter um botão que o leve para a home page localizada na home page.
- Permitir substituir elementos padrão para páginas particulares. Se, por exemplo, quisermos botões de navegação diferentes nas seções do site, devemos ser capazes de substituir os botões de navegação padrão.

Escrevendo o código para a classe

Se já decidimos a aparência do código a partir da saída e alguns recursos que gostaríamos que ele tivesse, como implementá-lo? Conversaremos mais adiante no livro sobre design e gerenciamento de projeto para projetos grandes. Por enquanto, nos concentraremos na parte específica de escrever PHP orientado a objetos.

Nossa classe precisará de um nome lógico. Como representa uma página, será chamada Page. Para declarar uma classe chamada Page, digitamos:

```
class Page
{
}
```

Nossa classe precisa de alguns atributos. Para esse exemplo, configurare elementos que talvez queira que sejam alterados de uma página para outra como atributos da classe. O conteúdo principal da página, que será uma combinação de tags HTML e texto, será chamado $content. Podemos declarar o conteúdo com a seguinte linha de código dentro da definição de classe:

```
public $content;
```

Também podemos configurar atributos para armazenar o título da página. Provavelmente vamos querer alterar isso para mostrar claramente que página particular nosso visitante está vendo. Em vez de ter títulos em branco, forneceremos um título padrão com a seguinte declaração:

```
public $title = 'TLA Consulting Pty Ltd';
```

A maioria das páginas comerciais Web inclui metatags para ajudar utilitários de pesquisa a indexá-las. Para serem úteis, as metatags provavelmente devem mudar de uma página para outra. Novamente, forneceremos um valor padrão:

```
public $keywords = 'TLA Consulting, Three Letter Abbreviation,
                   some of my best friends are search engines';
```

Os botões de navegação mostrados na página original na Figura 5.2 (veja o capítulo anterior) provavelmente devem permanecer os mesmos de uma página para outra para evitar que as pessoas se confundam, mas para alterá-los facilmente, também faremos um atributo para eles. Como pode haver um número variável de botões, utilizaremos um array e armazenaremos tanto o texto para o botão como o URL para o qual ele deve apontar.

```
public $buttons = array( 'Home'     => 'home.php',
                         'Contact'  => 'contact.php',
                         'Services' => 'services.php',
                         'Site Map' => 'map.php'
                       );
```

A fim de fornecer alguma funcionalidade, nossa classe também precisará de operações. Podemos começar fornecendo funções acessoras para configurar e obter os valores dos atributos que definimos.

```
public function __set($name, $value)
{
 $this->$name = $value;
}
```

A função __set( ) não contém verificação de erro (para sermos breves), mas essa capacidade pode ser facilmente acrescentada posteriormente, conforme requerida. Como é improvável que você solicite quaisquer desses valores de fora da classe, pode escolher não fornecer uma função __get( ), como fizemos aqui.

O propósito principal dessa classe é exibir uma página de HTML, então precisaremos de uma função. Chamamos a Display( ) da seguinte forma:

```
public function Display( )
  {
    echo "<html>\n<head>\n";
    $this -> DisplayTitle( );
    $this -> DisplayKeywords( );
    $this -> DisplayStyles( );
    echo "</head>\n<body>\n";
    $this -> DisplayHeader( );
    $this -> DisplayMenu($this->buttons);
    echo $this->content;
    $this -> DisplayFooter( );
    echo "</body>\n</html>\n";
  }
```

A função inclui algumas instruções echo simples para exibir HTML, mas consiste principalmente em chamadas para outras funções na classe. Como você provavelmente adivinhou em seus nomes, essas outras funções exibem partes da página.

Não é obrigatório dividir funções desse modo. Todas essas funções separadas poderiam simplesmente ser combinadas em uma função grande. Separamos essas funções por várias razões.

Cada função deve ter uma tarefa definida para realizar. Quanto mais simples for essa tarefa, mais fácil será escrever e testar a função. Não vá longe demais – se dividir seu programa em unidades pequenas demais, talvez seja difícil ler.

Utilizando herança, podemos sobrescrever operações. Podemos substituir uma grande função Display( ), mas é improvável que desejaremos alterar a maneira como a página inteira é exibida. Será muito melhor dividir a funcionalidade de exibição e algumas tarefas autocontidas e ser capaz de sobrescrever somente as partes que queremos alterar.

Nossa função Display chama DisplayTitle( ), DisplayKeywords( ), DisplayStyles( ), DisplayHeader( ), DisplayMenu( ) e DisplayFooter( ). Isso significa que precisamos definir essas operações. Podemos escrever operações ou funções nessa ordem lógica, chamando a operação ou função antes do código atual para a função. Em muitas outras linguagens, precisamos escrever a função ou operação antes de poder chamá-las. A maioria das operações é relativamente simples e precisa exibir alguma HTML e talvez o conteúdo dos atributos.

A Listagem 6.1 mostra a classe completa, que salvamos como page.inc para incluir ou requerer em outros arquivos.

Listagem 6.1 page.inc – A classe de página (Page) fornece uma maneira flexível e fácil de criar páginas da TLA

```
<?php
class Page
{
  // atributos da classe Page
  var $content;
  var $title = 'TLA Consulting Pty Ltd';
  var $keywords = 'TLA Consulting, Three Letter Abbreviation,
                   some of my best friends are search engines';
  var $buttons = array( 'Home'     => 'home.php',
                        'Contact'  => 'contact.php',
                        'Services' => 'services.php',
                        'Site Map' => 'map.php'
                      );

  // operações da classe Page

  function SetContent($newcontent)
  {
    $this->content = $newcontent;
  }

  function SetTitle($newtitle)
  {
    $this->title = $newtitle;
  }

  function SetKeywords($newkeywords)
  {
    $this->keywords = $newkeywords;
  }

  function SetButtons($newbuttons)
  {
    $this->buttons = $newbuttons;
  }

  function Display( )
  {
    echo "<html>\n<head>\n";
    $this -> DisplayTitle( );
    $this -> DisplayKeywords( );
    $this -> DisplayStyles( );
    echo "</head>\n<body>\n";
    $this -> DisplayHeader( );
    $this -> DisplayMenu($this->buttons);
    echo $this->content;
    $this -> DisplayFooter( );
```

Listagem 6.1 Continuação

```
    echo "</body>\n</html>\n";
  }

  function DisplayTitle( )
  {
    echo '<title> $this->title </title>';
  }

  function DisplayKeywords( )
  {
    echo "<META name=\"keywords\" content=\"$this->keywords\">";
  }

  function DisplayStyles( )
  {
?>
  <style>
    h1 {color:white; font-size:24pt; text-align:center;
        font-family:arial,sans-serif}
    .menu {color:white; font-size:12pt; text-align:center;
          font-family:arial,sans-serif; font-weight:bold}
    td {background:black}
    p {color:black; font-size:12pt; text-align:justify;
       font-family:arial,sans-serif}
    p.foot {color:white; font-size:9pt; text-align:center;
            font-family:arial,sans-serif; font-weight:bold}
    a:link,a:visited,a:active {color:white}
  </style>
<?php
  }

  function DisplayHeader( )
  {
?>
  <table width="100%" cellpadding ="12" cellspacing ="0" border ="0">
  <tr bgcolor ="black">
    <td align ="left"><img src = "logo.gif"></td>
    <td>
        <h1>TLA Consulting Pty Ltd</h1>
    </td>
    <td align ="right"><img src = "logo.gif"></td>
  </tr>
  </table>
<?php
  }

  function DisplayMenu($buttons)
  {
    echo "<table width='100%' bgcolor='white' cellpadding='4'
              cellspacing='4'\n";
    echo "  <tr>\n";

    //calcula o tamanho do botão
    $width = 100/count($buttons);

    while (list($name, $url) = each($buttons))
    {
      $this -> DisplayButton($width, $name, $url, !$this->IsURLCurrentPage($url));
    }
    echo "  </tr>\n";
```

Listagem 6.1 Continuação

```
    echo "</table>\n";
  }

  function IsURLCurrentPage($url)
  {
    if(strpos( $GLOBALS['SCRIPT_NAME'], $url )==false)
    {
      return false;
    }
    else
    {
      return true;
    }
  }

  function DisplayButton($width, $name, $url, $active = true)
  {
    if ($active)
    {
      echo "<td width ='$width%'>
           <a href ='$url'>
            <img src ='s-logo.gif' alt ='$name' border ='0'></a>
            <a href ='$url'><span class='menu'>$name</span></a></td>";
    }
    else
    {
      echo "<td width ='$width%'>
            <img src ='side-logo.gif'>
            <span class='menu'>$name</span></td>";
    }
  }

  function DisplayFooter( )
  {
?>
    <table width = "100%" bgcolor ="black" cellpadding ="12" border ="0">
    <tr>
      <td>
        <p class="foot">&copy; TLA Consulting Pty Ltd.</p>
        <p class="foot">Please see our
                      <a href ="">legal information page</a></p>
      </td>
    </tr>
    </table>
<?php
  }
}
?>
```

Ao ler essa classe, note que DisplayStyles( ), DisplayHeader( ) e DisplayFooter( ) precisam exibir um bloco grande de HTML estática, sem processamento de PHP. Portanto, utilizamos simplesmente uma tag de PHP de fechamento (?>), digitamos a HTML e então reinserimos o PHP com uma tag de PHP de abertura (<?php) enquanto dentro das funções.

Duas outras operações são definidas nessa classe. A saída da operação DisplayButton( ) é um botão de menu simples. Se o botão estiver apontando para a página em que estamos, em vez disso estamos exibindo um botão inativo, que é ligeiramente diferente e não se vincula a lugar nenhum. Isso mantém o layout de página consistente e fornece aos visitantes uma localização visual.

A operação IsURLCurrentPage( ) determina se o URL para um botão aponta para a página atual. Uma grande quantidade de técnicas pode ser utilizada para descobrir isso. Utilizamos a função de string strpos( ) para ver se o URL fornecido está contido em uma das variáveis configuradas de servidor. A instrução strpos($GLOBALS['SCRIPT_NAME'], $url) retornará um número se a string em $url estiver dentro da variável global SCRIPT_NAME ou retornará false se não estiver.

Para utilizar essa classe de página, precisamos incluir page.inc em um script e chamar Display( ).

O código na Listagem 6.2 criará a home page TLA Consulting e fornecerá saída muito semelhante àquela que anteriormente geramos na Figura 5.2. O código na Listagem 6.2 faz o seguinte:

1. Utiliza require para incluir o conteúdo da page.inc, que contém a definição da classe Page.
2. Cria uma instância da classe Page. A instância é chamada $homepage.
3. Chama a operação SetContent( ) dentro do objeto $homepage e passa algum texto e tags HTML para aparecer na página.
4. Chama a operação Display( ) dentro do objeto $homepage para fazer com que a página seja exibida no navegador do visitante.

Listagem 6.2 **home.php – Essa home page utiliza a classe Page para fazer a maior parte do trabalho envolvido na geração da página**

```
<?php
  require ('page.inc');

  $homepage = new Page( );

  $homepage -> SetContent('<p>Welcome to the home of TLA Consulting.
                          Please take some time to get to know us.</p>
                          <p>We specialize in serving your business needs
                          and hope to hear from you soon.</p>'
                         );
  $homepage -> Display( );
?>
```

Você pode ver na Listagem 6.2 que precisamos trabalhar muito pouco para gerar novas páginas utilizando essa classe Page. Utilizar a classe dessa maneira significa que todas as páginas precisam ser muito semelhantes.

Se quisermos que algumas seções do site utilizem uma variante da página padrão, simplesmente poderíamos copiar page.inc para um novo arquivo chamado page2.inc e fazer algumas alterações. Isso significaria que toda vez que atualizássemos ou corrigíssemos partes da page.inc, precisaríamos lembrar de fazer as mesmas alterações na page2.inc.

Um curso melhor de ação é utilizar a herança para criar uma nova classe que herda a maior parte da sua funcionalidade a partir de Page, mas sobrescreve as partes que precisam ser diferentes. Para o site da TLA, requeremos que a página de serviços inclua uma segunda barra de navegação. O script mostrado na Listagem 6.3 faz isso criando uma nova classe chamada ServicesPage que herda de Page. Fornecemos um novo array chamado $row2buttons que contém os botões e links que queremos na segunda linha. Como desejamos que essa classe comporte-se da mesma maneira em sua maior parte, só anulamos a parte que queremos que seja alterada – a operação Display( ).

Listagem 6.3 **services.php** – A página de serviços herda da classe **Page** mas sobrescreve **Display( )** para alterar a saída

```
<?php
  require ('page.inc');

  class ServicesPage extends Page
  {
    private $row2buttons = array( 'Re-engineering' => 'reengineering.php',
                                  'Standards Compliance' => 'standards.php',
                                  'Buzzword Compliance' => 'buzzword.php',
                                  'Mission Statements' => 'mission.php'
                                );
    public function Display( )
    {
      echo "<html>\n<head>\n";
      $this -> DisplayTitle( );
      $this -> DisplayKeywords( );
      $this -> DisplayStyles( );
      echo "</head>\n<body>\n";
      $this -> DisplayHeader( );
      $this -> DisplayMenu($this->buttons);
      $this -> DisplayMenu($this->row2buttons);
      echo $this->content;
      $this -> DisplayFooter( );
      echo "</body>\n</html>\n";
    }
  }

  $services = new ServicesPage( );
  $services -> content ='<p>At TLA Consulting, we offer a number of services.
          Perhaps the productivity of your employees would
          improve if we re-engineered your business.
          Maybe all your business needs is a fresh mission
          statement, or a new batch of buzzwords.</p>';
  $services -> SetContent($content);
  $services -> Display( );
?>
```

A `Display( )` substituta é muito semelhante, mas contém a linha extra

```
$this -> DisplayMenu($this->row2buttons);
```

para chamar `DisplayMenu( )` uma segunda vez e criar uma segunda barra de menu.

Fora da definição de classe, criamos uma instância da classe `ServicesPage`, configuramos os valores para os quais queremos valores não-padrão e chamamos `Display( )`.

Como mostrado na Figura 6.2, temos uma nova variante da página padrão. O único código novo que precisamos escrever foi para as partes que eram diferentes.

Criar páginas via classes de PHP tem vantagens óbvias. Com uma classe que faz a maior parte do trabalho para nós, temos menos trabalho para criar uma nova página. Podemos atualizar todas as páginas de uma vez simplesmente atualizando a classe. Utilizando herança, podemos derivar versões diferentes da classe de nosso original sem comprometer as vantagens.

Como na vida, essas vantagens não vêm de graça. Criar páginas a partir de um script requer mais esforço do processador do computador que simplesmente carregar uma página HTML estática a partir do disco e enviá-la para um navegador. Em um site ocupado, isso será importante e você deve fazer um esforço tanto para utilizar páginas HTML estáticas como para armazenar em cache a saída de seus scripts onde for possível para reduzir a carga no servidor.

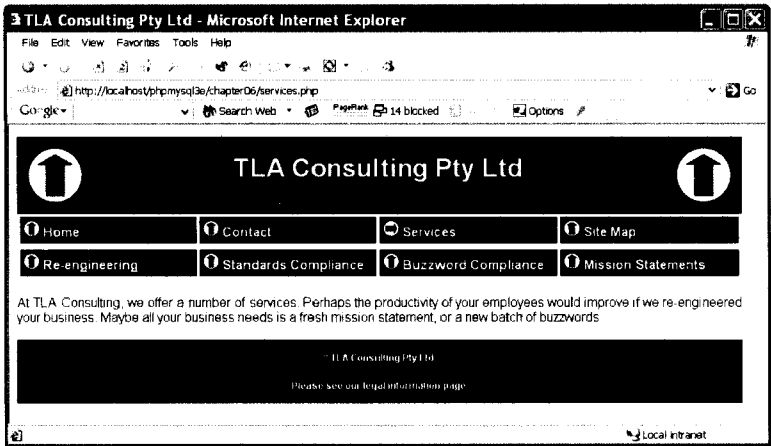



Figura 6.2 A página de serviços é criada utilizando herança para reutilizar a maior parte da página padrão.

Compreendendo a nova funcionalidade avançada de orientação a objetos no PHP

Nas seções seguintes, discutiremos os recursos avançados de orientação a objetos do PHP, a maior parte é nova no PHP5.

Nota: PHP4 *versus* PHP5

Se você tiver utilizado versões anteriores do PHP, é preciso conhecer algumas diferenças importantes.

No PHP4, os objetos eram passados por valor, agora são passados por referência. Essa escrita de código não deve quebrar qualquer código antigo, e, na verdade, muitos programadores escreviam código ineficiente sem perceber. Por exemplo, escrever

```
$c = new myClass;
```

criava uma nova classe e então copiava a instância para $c (criando duas cópias, mas perdendo imediatamente o handle para uma delas). Isso pode gerar problemas se você considerar que objetos estão sendo passados por referência, especialmente quando você está passando objetos a funções.

A maioria das linguagens orientadas a objetos passa objetos como referências por padrão, e agora o PHP se inclui nessa categoria.

A outra diferença importante é que o PHP anteriormente tinha dificuldade em tirar a referência de objetos que eram retornados de funções, normalmente para chamar métodos naqueles objetos. Antes, não era possível escrever, por exemplo,

```
select_object( )->display( );
```

onde `select_object( )` retornava um objeto que tinha um método chamado `display( )`. Essa instrução agora deve funcionar sem problemas.

Utilizando constantes por classe

Constante por classe também é uma novidade no PHP5. Essa constante pode ser utilizada sem precisar instanciar a classe, como neste exemplo:

```
<?php
class Math {
  const pi = 3.14159;
}
```

```
echo 'Math::pi = '.Math::pi."\n";
?>
```

Você pode acessar a constante por classe utilizando o operador :: para especificar a classe à qual pertence a constante, como foi feito nesse exemplo.

Implementando métodos estáticos

O PHP5 também introduz a palavra-chave `static`. Ela é aplicada a métodos que permitem que sejam chamados sem instanciar a classe, semelhante à idéia das constantes de classe. Por exemplo, considere a classe `Math` criada na seção anterior. Você poderia adicionar uma função `squared( )` a ela e chamá-la sem instanciar a classe, desta forma:

```
class Math
{
static function squared($input)
{
  return $input*$input;
}
}
echo Math::squared(8);
```

Observe que não é possível utilizar a palavra-chave `this` dentro de um método estático porque pode não haver instância de objeto à qual fazer referência.

Verificando o tipo de classe e Type Hinting

Outras novidades no PHP5 são a palavra-chave `instanceof` e o conceito de type hinting.

A palavra-chave `instanceof` permite verificar o tipo de um objeto. Você pode verificar se um objeto é uma instância de uma classe específica, se herda de uma classe ou se implementa uma interface. A palavra-chave `instanceof` é efetivamente um operador condicional. Por exemplo, com os exemplos anteriores em que implementamos a `B` como subclasse da classe `A`, então:

`($b instanceof B)`	seria verdadeiro.
`($b instanceof A)`	seria verdadeiro.
`($b instanceof Displayable)`	seria falso.

Todos esses exemplos supõem que `A`, `B` e `Displayable` estão no escopo atual; caso contrário, surgirá um erro.

O conceito de dica de tipo (*type hinting*) de classe também é novo no PHP5. Normalmente, ao passar um parâmetro a uma função no PHP, você não passa o tipo daquele parâmetro. Com o *type hinting*, você pode especificar o tipo da classe que deve ser passado, e se não for o tipo passado realmente, um erro será gerado. A verificação de tipo é equivalente a `instanceof`. Por exemplo, considere a seguinte função:

```
function check_hint(B $someclass)
{
 //...
}
```

Esse exemplo sugere que `$someclass` precisa ser uma instância da classe `B`. Se você então passar uma instância da classe `A` como

```
check_hint($a);
```

receberá o seguinte erro fatal:

```
Fatal error: Argument 1 must be an instance of B
```

Observe que se você tivesse inserido a dica em A e passado uma instância de B, não ocorreria nenhum erro porque B herda de A.

Clonando objetos

A palavra-chave clone também é nova no PHP5, e permite copiar um objeto existente. Por exemplo,

```
$c = clone $b;
```

criaria uma cópia do objeto $b da mesma classe, com os mesmos valores de atributos.

Você também pode mudar esse comportamento. Se precisar de um comportamente não-default de clone, deve criar um método na classe base chamado __clone( ). Esse método é semelhante ao construtor ou destruidor, no sentido de que você não o chama diretamente. Ele é chamado quando a palavra-chave clone é usada como mostrado aqui. Dentro do método __clone( ), você então pode definir exatamente o comportamente que deseja copiar.

Uma boa coisa a respeito de __clone( ) é que ele será chamado depois que uma cópia exata tiver sido feita usando o comportamento padrão, então, naquele estágio, você pode alterar apenas o que deseja.

A funcionalidade mais comum a acrescentar a __clone( ) é código para garantir que os atributos de uma classe tratados como referências sejam copiados corretamente. Se você definir clonar uma classe que contenha referência a um objeto, provavelmente está esperando uma segunda cópia daquele objeto, em vez de uma segunda referência ao mesmo; portanto, seria útil adicioná-lo a __clone( ).

Você também pode escolher não fazer mudanças, e realizar outra ação, como atualizar um registro de banco de dados relacionado à classe.

Utilizando classes abstratas

Classes abstratas são um outro novo recurso no PHP5. Elas não podem ser instanciadas.

O PHP5 também oferece métodos abstratos, que fornecem assinatura para um método, mas nenhuma implementação, como neste exemplo:

```
abstract operationX($param1, $param2);
```

Qualquer classe que contenha métodos abstratos também deve ser abstrata, como mostra este exemplo:

```
abstract class A
{
 abstract function operationX($param1, $param2);
}
```

O principal uso de classes e métodos abstratos é em uma hierarquia complexa de classes, na qual se deseja garantir que cada subclasse contém e sobrescreve algum método específico; isso também pode ser feito com interface.

Sobrecarregando métodos com __call( )

Anteriormente, vimos alguns métodos de classes com significados especiais, cujos nomes começam com um sublinhado duplo (__), como __get( ), __set( ), __construct( ) e __destruct( ). Outro exemplo é o método __call( ), que é usado no PHP para implementar sobrecarga de método.

A sobrecarga de método é comum em muitas linguagens orientadas a objeto, mas não é tão útil no PHP porque a tendência é usar tipos flexíveis e os parâmetros de função opcionais (fáceis de implementar).

Para utilizá-la, é preciso implementar o método __call( ), como neste exemplo:

```
public function __call($method, $p)
{
 if ($method == 'display')
   if (is_object($p[0]))
      $this->displayObject($p[0]);
   else if (is_array($p[0]))
      $this->displayArray($p[0]);
   else
      $this->displayScalar($p[0]);
}
```

O método `__call( )` deve usar dois parâmetros. O primeiro contém o nome do método sendo chamado, e o segundo contém um array dos parâmetros passados ao método. Então, você pode decidir sozinho que método subjacente chamar. Nesse caso, se um objeto é passado ao método `display( )`, é preciso chamar o método `displayObject( )`; se um array é passado, chame `displayArray( )`; e se algo mais for passado, chame `displayScalar( )`.

Para chamar esse código, primeiro instancie a classe contendo o método `__call( )` (nomeie-a como `overload`) e então chame o método `display( )`, como neste exemplo:

```
$ov = new overload;
$ov->display(array(1, 2, 3));
$ov->display('cat');
```

A primeira chamada a `display( )` é `displayArray( )` e a segunda, `displayScalar( )`.

Observe que você não precisa de nenhuma implementação do método `display( )` para que o código funcione.

Utilizando __autoload( )

Outra função especial é `_autoload( )`. Ela não é um método de classe, e sim uma função independente, ou seja, você a declara de fora de qualquer declaração de classe. Se implementá-la, ela será automaticamente chamada quando você tentar instanciar uma classe que não tenha sido declarada.

O principal uso de `__autoload( )` é para tentar incluir ou exigir arquivos necessários para instanciar a classe requerida. Considere este exemplo:

```
function __autoload($name)
{
  include_once $name.'.php';
}
```

Essa implementação tenta incluir um arquivo com o mesmo nome que a classe.

Implementando iteradores e iteração

Um recurso inteligente da nova capacidade de orientação a objetos é que você pode utilizar um loop `foreach( )` para iterar os atributos de um objeto, como faria com um array. Aqui está um exemplo:

```
class myClass
{
 public $a=5;
 public $b=7;
 public $c=9;
}
$x = new myClass;
foreach ($x as $attribute)
 echo $attribute.'<br />';
```

(Enquanto escrevíamos este livro, o manual do PHP ainda sugeria que é necessário implementar a interface vazia de `Traversable` para que `foreach` funcione, mas isso gera um erro fatal. No entanto, não implementá-la parece funcionar bem.)

Se precisar de algum comportamento mais sofisticado do que esse, você pode implementar um iterador. Para isso, faça com que a classe que deseja iterar implemente a interface `IteratorAggregate` e forneça a ela um método `getIterator` que retorne uma instância da classe iteradora. A classe deve implementar a interface `Iterator`, que possui uma série de métodos que devem ser implementados. Um exemplo de classe e de iterador é mostrado na Listagem 6.4.

Listagem 6.4 **iterator.php – Um exemplo de classe base e classe iteradora**

```
<?php
class ObjectIterator implements Iterator {

  private $obj;
  private $count;
  private $currentIndex;

  function __construct($obj)
  {
    $this->obj = $obj;
    $this->count = count($this->obj->data);
  }
  function rewind( )
  {
    $this->currentIndex = 0;
  }
  function valid( )
  {
    return $this->currentIndex < $this->count;
  }
  function key( )
  {
    return $this->currentIndex;
  }
  function current( )
  {
    return $this->obj->data[$this->currentIndex];
  }
  function next( )
  {
    $this->currentIndex++;
  }
}

class Object implements IteratorAggregate
{
 public $data = array( );

 function __construct($in)
 {
   $this->data = $in;
 }

 function getIterator( )
 {
   return new ObjectIterator($this);
 }
}

$myObject = new Object(array(2, 4, 6, 8, 10));

$myIterator = $myObject->getIterator( );
```

Listagem 6.4 Continuação

```
for($myIterator->rewind( ); $myIterator->valid( ); $myIterator->next( ))
{
 $key = $myIterator->key( );
 $value = $myIterator->current( );
 echo "$key => $value <br />";
}
?>
```

A classe ObjectIterator possui um conjunto de funções, conforme exigido pela interface Iterator:

- O construtor não é requerido, mas obviamente é um bom local para definir valores para o número de itens nos quais você planeja iterar e um link ao item de dados atual.
- A função rewind( ) deve definir o ponteiro de dados internos de volta ao início dos dados.
- A função valid( ) deve informar se existem mais dados no local atual do ponteiro de dados.
- A função key( ) deve retornar o valor do ponteiro de dados.
- A função value( ) deve retornar o valor armazenado no ponteiro de dados atual.
- A função next( ) deve mover o ponteiro pelos dados.

A razão para usar uma classe iteradora é que a interface para os dados não mudará mesmo se a implementação mudar. Nesse exemplo, a classe IteratorAggregate é um array simples. Se você decidir mudá-la para uma tabela hash ou lista vinculada, pode utilizar um Iterator padrão para atravessá-la, embora o código Iterator seja diferente.

Convertendo as classes em strings

Outra das novas funções "mágicas" é __toString( ). Se implementar uma função chamada __toString( ) em sua classe, ela será chamada quando você tentar imprimir a classe, como neste exemplo:

```
$p = new Printable;
echo $p;
```

O que a função __toString( ) retornar será impresso por echo. Você pode implementá-la, por exemplo, assim:

```
class Printable
{
 var $testone;
 var $testtwo;
 public function __toString( )
 {
   return(var_export($this, TRUE));
 }
}
```

(A função var_export( ) imprime todos os valores de atributos na classe.)

Utilizando a Reflection API

Um novo recurso interessante no PHP5 é a *reflection API*. *Reflection* é a capacidade de examinar classes e objetos existentes para descobrir sobre sua estrutura e conteúdo. Essa capacidade pode ser útil quando você está lidando com classes desconhecidas ou não-documentadas, como em scripts PHP codificados.

A API é extremamente complexa, mas veremos um exemplo simples que dará uma idéia de seu uso. Considere a classe `Page` definida neste capítulo, por exemplo. Você pode obter toda informação sobre essa classe na Reflection API, como mostra a Listagem 6.5.

Listagem 6.5 **`reflection.php`** – **Exibe informações sobre a classe `page`**

```
<?php
require_once('page.inc');
$class = new ReflectionClass('Page');
echo '<pre>';
echo $class;
echo '</pre>';
?>
```

Aqui, você utiliza o método `__toString( )` da classe `Reflection class` para imprimir esses dados. Obverve que as tags <pre> estão em linhas separadas para não confundir com o método `__toString( )`.

A primeira tela desse código é mostrada na Figura 6.3.

Figura 6.3 A saída da Reflection API é surpreendentemente detalhada.

A seguir

O próximo capítulo explica as novas capacidades de lidar com exceções do PHP. As exceções fornecem um mecanismo elegante para lidar com erros de tempo de execução.

7

Tratamento de exceções

NESTE CAPÍTULO, explicamos o conceito de tratamento de exceções e a forma como é implementado no PHP. Exceções são um novo e importante recurso no PHP5. Elas oferecem um mecanismo unificado para manipular erros de uma forma extensiva, sustentável e orientada a objetos.

Os principais tópicos abrangidos neste capítulo incluem:

- Conceitos de tratamento de exceções;
- Estruturas de controle de exceções: `try...throw...catch`;
- A classe `Exception`;
- Exceções definidas pelo usuário;
- Exceções no exemplo Bob's Auto Parts;
- Exceções e outros mecanismos de manipulação de erros do PHP.

Conceitos de tratamento de exceções

A idéia básica do tratamento de exceções é que o código é executado dentro do chamado bloco `try`, uma seção de código que se parece com isto:

```
try
{
 // o código entra aqui
}
```

Se algo estiver errado dentro do bloco `try`, você pode fazer o que chamamos de lançar uma exceção. Algumas linguagens, como Java, lançam exceções automaticamente em certos casos. No PHP, as exceções devem ser lançadas manualmente, desta forma:

```
throw new Exception('mensagem', código);
```

A palavra-chave `throw` ativa o mecanismo de tratamento de exceções. É uma construção de linguagem, e não uma função, mas é preciso passar um valor a ela, que espera receber um objeto. No caso mais simples, você pode instanciar a classe embutida `Exception`, como feito nesse exemplo.

O construtor dessa classe utiliza dois parâmetros: uma mensagem e um código. Eles devem representar uma mensagem de erro e um número de código de erro. Ambos os parâmetros são opcionais.

Finalmente, no bloco `try`, você precisa de, pelo menos, um bloco `catch`, que se parece com isto:

```
catch (typehint exception)
{
 // manipula exceção
}
```

Você pode ter mais de um bloco catch associado a um único bloco try. Usar mais de um faz sentido se cada bloco catch estiver esperando pegar um tipo diferente de exceção. Por exemplo, se você quiser pegar exceções da classe Exception, o bloco catch pode se parecer com isto:

```
catch (Exception $e)
{
 // manipula exceção
}
```

O objeto passado para (e pego pelo) o bloco catch é aquele passado para a (e lançado pela) instrução throw que gerou a exceção. A exceção pode ser de qualquer tipo, mas é bom usar instâncias da classe Exception ou instâncias das próprias exceções definidas pelo usuário, que herdam da classe Exception. (Você verá como definir suas próprias exceções posteriormente neste capítulo.)

Quando uma exceção é gerada, o PHP procura por um bloco catch correspondente. Se tiver mais de um bloco catch, os objetos passados a cada um devem ser de tipos diferentes, de modo que o PHP possa achar em que bloco catch atuar.

Um outro ponto a observar é que você pode levantar mais exceções dentro de um bloco catch.

Para esclarecer essa discussão um pouco mais, vamos ver um exemplo simples de tratamento de exceção, mostrado na Listagem 7.1.

Listagem 7.1 **basic_exception.php – Lançando e capturando uma exceção**

```
<?php

try
{
 throw new Exception('A terrible error has occurred', 42);
}
catch (Exception $e)
{
 echo 'Exception '. $e->getCode( ). ': '. $e->getMessage( )
      .' in '. $e->getFile( ). ' on line '. $e->getLine( ). '<br />';
}
?>
```

Na Listagem 7.1, você pode ver que utilizamos alguns métodos da classe Exception, que discutimos rapidamente. O resultado da execução desse código é mostrado na Figura 7.1.

No código de exemplo, você pode ver que levantamos uma exceção da classe Exception. Essa classe embutida possui métodos que você pode utilizar no bloco catch para transmitir uma mensagem útil de erro.

A classe Exception

O PHP5 vem com uma classe embutida chamada Exception. O construtor utiliza dois parâmetros, conforme discutimos anteriormente: uma mensagem de erro e um código de erro.

Além do construtor, essa classe vem com os seguintes métodos:

- getCode( ) – Retorna o código conforme passado ao construtor.
- getMessage( ) – Retorna a mensagem conforme passada ao construtor.

Figura 7.1 Esse bloco `catch` apresenta uma mensagem de erro de exceção e destaca onde ocorreu.

- `getFile( )` – Retorna o caminho completo para o arquivo de código onde a exceção foi gerada.
- `getLine( )` – Retorna o número da linha no arquivo de código onde a exceção foi gerada.
- `getTrace( )` – Retorna um array contendo um backtrace onde a exceção foi gerada.
- `getTraceAsString( )` – Retorne a mesma informação que `getTrace`, formatada como string.
- `__toString( )` – Permite simplesmente ecoar um objeto `Exception`, dando todas as informações dos métodos anteriores.

Você pode ver que utilizamos os quatro primeiros métodos na Listagem 7.1. Você poderia obter a mesma informação (mais o backtrace) executando:

```
echo $e;
```

Um *backtrace* mostra que funções estavam sendo executadas no momento em que a exceção foi gerada.

Exceções definidas pelo usuário

Em vez de instanciar e passar uma instância da classe base `Exception`, você pode passar qualquer outro objeto desejado. Na maioria dos casos, você partirá da classe `Exception` para criar suas próprias classes de exceção.

É possível passar qualquer outro objeto com a cláusula `throw`. Você pode occasionalmente querer fazer isso se estiver tendo problemas com um objeto específico e deseja passá-lo por propósitos de depuração.

Entretanto, na maior parte das vezes, você partirá da classe base `Exception`. O manual do PHP fornece um código que mostra o formato da classe `Exception`. Esse código, obtido em http://www.php.net/zend-engine-2.php, é reproduzido na Listagem 7.2. Observe que este não é o código real, mas representa o que você pode esperar herdar.

Listagem 7.2 **Classe `Exception` – Isso é o que você pode esperar herdar**

```
<?php
class Exception {
  function __construct(string $message=NULL, int $code=0) {
      if (func_num_args( )) {
          $this->message = $message;
      }
```

Listagem 7.2 **Continuação**

```
        $this->code = $code;
        $this->file = __FILE__; // da cláusula throw
        $this->line = __LINE__; // da cláusula throw
        $this->trace = debug_backtrace( );
        $this->string = StringFormat($this);
    }

    protected $message = 'Unknown exception'; // mensagem de exceção
    protected $code = 0; // código de exceção definido pelo usuário
    protected $file;    // nome de arquivo de origem da exceção
    protected $line;    // linha de origem da exceção

    private $trace;      // backtrace da exceção
    private $string;    // interno apenas!!

    final function getMessage( ){
        return $this->message;
    }
    final function getCode( ) {
        return $this->code;
    }
    final function getFile( ) {
        return $this->file;
    }
    final function getTrace( ) {
        return $this->trace;
    }
    final function getTraceAsString( ) {
        return self::TraceFormat($this);
    }
    function _toString( ) {
        return $this->string;
    }
    static private function StringFormat(Exception $exception) {
        // ... uma função não disponível nos scripts do PHP
        // que retorna todas as informações relevantes como string
    }
    static private function TraceFormat(Exception $exception) {
        // ... uma função não disponível nos scripts do PHP
        // que retorna todas as informações relevantes como string
    }
}
? >
```

A principal razão pela qual estamos examinando a definição dessa classe é para destacar que todos os métodos públicos são finais: isso significa que não é possível sobrescrevê-los. Você pode criar sua própria subclasse `Exceptions`, mas não pode mudar o comportamento dos métodos básicos. Observe que como é possível sobrescrever a função `__toString( )`, a forma como a exceção é exibida também pode ser modificada. Você também pode acrescentar seus próprios métodos.

Um exemplo de uma classe `Exception` definida por usuário é mostrado na Listagem 7.3.

Listagem 7.3 **`user_defined_exception.php` – Um exemplo de uma classe `Exception` definida pelo usuário**

```
<?php

class myException extends Exception
```

Listagem 7.3 Continuação

```
{
 function __toString( )
 {
      return '<table border><tr><td><strong>Exception '. $this->getCode( )
            . '</strong>: '. $this->getMessage( ).'<br />'.' in '
            . $this->getFile( ). ' on line '. $this->getLine( )
            . '</td></tr></table><br />';
  }
}

try
{
 throw new myException('A terrible error has occurred', 42);
}
catch (myException $m)
{
  echo $m;
}

?>
```

Nesse código, declaramos uma nova classe de exceção, chamada `myException`, que amplia a classe básica `Exception`. A diferença entre essa classe e a classe `Exception` é que você sobrescreve o método `__toString( )` para fornecer uma maneira "bonita" de imprimir a exceção. A saída da execução dessa código é mostrada na Figura 7.2.

Figura 7.2 A classe `myException` fornece exceções com uma "impressão bonita".

Esse exemplo é bastante simples. Na próxima seção, veremos maneiras de criar exceções diferentes para lidar com diferentes categorias de erros.

Exceções no exemplo Bob's Auto Parts

O Capítulo 2 descreveu como você poderia armazenar dados de pedido do Bob em um arquivo simples. Você sabe que a E/S de um arquivo (na verdade, qualquer tipo de E/S) é um local nos programas em que os erros ocorrem com freqüência, o que a torna um bom local para aplicar o tratamento de exceções.

Voltando ao código original, você pode ver que há três coisas que podem dar errado ao gravar para o arquivo: ele não pode ser aberto, um bloqueio não pode ser obtido ou o arquivo não pode ser

gravado. Criamos uma classe de exceção para cada uma dessas possibilidades. O código para essas exceções é mostrado na Listagem 7.4.

Listagem 7.4 **file_exceptions.php – Exceções relacionadas à E/S do arquivo**

```
<?php

class fileOpenException extends Exception
{
 function __toString( )
 {
      return 'fileOpenException '. $this->getCode( )
             . ': '. $this->getMessage( ).'<br />'.' in '
             . $this->getFile( ). ' on line '. $this->getLine( )
             . '<br />';
  }
}

class fileWriteException extends Exception
{
 function __toString( )
 {
      return 'fileWriteException '. $this->getCode( )
             . ': '. $this->getMessage( ).'<br />'.' in '
             . $this->getFile( ). ' on line '. $this->getLine( )
             . '<br />';
  }
}

class fileLockException extends Exception
{
 function __toString( )
 {
      return 'fileLockException '. $this->getCode( )
             . ': '. $this->getMessage( ).'<br />'.' in '
             . $this->getFile( ). ' on line '. $this->getLine( )
             . '<br />';
  }
}

?>
```

Essas subclasses Exception não fazem nada particularmente interessante. Na verdade, para essa aplicação, você pode deixá-las como subclasses vazias ou usar a classe fornecida Exception. Entretanto, fornecemos um método __toString( ) para cada uma das subclasses que explica que tipo de exceção ocorreu.

Reescrevemos o arquivo processorder.php do Capítulo 2 para incorporar o uso de exceções. A nova versão é mostrada na Listagem 7.5.

Listagem 7.5 **processorder.php – O script de processamento de pedidos do Bob com o tratamento de exceções incluído**

```
<?php

 require_once('file_exceptions.php');

 // cria nomes abreviados de variáveis
 $tireqty = $_POST['tireqty'];
```

Listagem 7.5 **Continuação**

```
 $oilqty = $_POST['oilqty'];
 $sparkqty = $_POST['sparkqty'];
 $address = $_POST['address'];

 $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
 <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<?php
$date = date('H:i, jS F');

echo '<p>Order processed at ';
echo $date;
echo '</p>';

echo '<p>Your order is as follows: </p>';

$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br />';

if( $totalqty == 0)
{
 echo 'You did not order anything on the previous page!<br />';
}
else
{
 if ( $tireqty>0 )
   echo $tireqty.' tires<br />';
 if ( $oilqty>0 )
   echo $oilqty.' bottles of oil<br />';
 if ( $sparkqty>0 )
   echo $sparkqty.' spark plugs<br />';
}

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
            + $oilqty * OILPRICE
            + $sparkqty * SPARKPRICE;

$totalamount=number_format($totalamount, 2, '.', ' ');

echo '<p>Total of order is '.$totalamount.'</p>';
echo '<p>Address to ship to is '.$address.'</p>';

$outputstring = $date."\t".$tireqty." tires \t".$oilqty." oil\t"
                 .$sparkqty." spark plugs\t\$".$totalamount
                 ."\t". $address."\n";

// abre arquivo para anexar
```

Listagem 7.5 Continuação

```
try
{
 if (!($fp = @fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab')))
     throw new fileOpenException( );

 if (!flock($fp, LOCK_EX))
    throw new fileLockException( );

 if (!fwrite($fp, $outputstring, strlen($outputstring)))
    throw new fileWriteException( );
 flock($fp, LOCK_UN);
 fclose($fp);
 echo '<p>Order written.</p>';
}
catch (fileOpenException $foe)
{
  echo '<p><strong>Orders file could not be opened. '
       .'Please contact our webmaster for help.</strong></p>';
}
catch (Exception $e)
{
  echo '<p><strong>Your order could not be processed at this time. '
       .'Please try again later.</strong></p>';
}

?>
</body>
</html>
```

Você pode ver que a seção E/S do arquivo no script está dentro de um bloco `try`. Geralmente é uma boa prática de codificação criar pequenos blocos `try` e pegar as exceções relevantes no final de cada um. Isso torna o código de tratamento de exceções mais fácil de escrever e manter, porque você pode ver com o que está lidando.

Se não puder abrir o arquivo, lance uma `fileOpenException`; se não puder bloquear o arquivo, lance uma `fileLockException`; e se não puder gravar no arquivo, lance uma `fileWriteException`.

Observe os blocos `catch blocks`. Para ilustrar um ponto, incluímos apenas dois: um para manipular `fileOpenExceptions` e outro para lidar com `Exception`. Como as outras exceções herdam de `Exception`, elas serão capturadas pelo segundo bloco `catch`. Os blocos `Catch` são comparados na mesma base que o operador `instanceof`. Essa é uma boa razão para estender suas classes de exceção a partir de uma única classe.

Um aviso importante: se você levantar uma exceção para a qual não escreveu um bloco `catch`, o PHP reportará um erro fatal.

Exceções e outros mecanismos de tratamento de erros do PHP

Além do mecanismo de tratamento de exceções discutido neste capítulo, o PHP possui um complexo suporte a manipulação de erros, que veremos no Capítulo 25. Observe que o processo de levantar e tratar exceções não interfere ou evita que o mecanismo de tratamento de erros opere.

Na Listagem 7.5, observe como a chamada a `fopen( )` ainda é precedida por um operador de supressão de erros `@`. Se falhar, o PHP gerará um aviso que pode ou não ser notificado, dependendo das configurações de notificação de erros em `php.ini`. Essas configurações são discutidas em detalhes no Capítulo 25, mas você precisa saber que esse aviso será lançado, independente de você ter levantado ou não uma exceção.

Leitura adicional

Como o tratamento de exceções é novo no PHP, não há muita literatura sobre o assunto. Entretanto, há muitas informações básicas disponíveis. A Sun possui um bom tutorial sobre o que são exceções e por que você pode querer utilizá-las (escrito a partir de uma perspectiva da Java, é claro), localizado em http://java.sun.com/docs/books/tutorial/essential/_exceptions/definition.html.

A seguir

A próxima parte do livro lida com o MySQL. Explicamos como criar e preencher um banco de dados MySQL para então relacionar ao que aprendeu sobre PHP de modo que você possa acessar o banco de dados a partir da Web.

II

Utilizando MySQL

8

Projetando bancos de dados Web

AGORA QUE VOCÊ ESTÁ FAMILIARIZADO com os princípios básicos do PHP, começaremos a ver a integração de um banco de dados nos seus scripts. Como você pode lembrar, no Capítulo 2, conversamos sobre as vantagens de utilizar um banco de dados relacional em vez de um arquivo simples (*flat file*). Elas incluem:

- Os RDBMSs podem fornecer acesso mais rápido aos dados que aos arquivos simples (flat file).
- Os RDBMSs podem ser facilmente consultados para extrair conjuntos de dados que satisfaçam certos critérios.
- Os RDBMSs têm mecanismos predefinidos para lidar com acesso concorrente para que você, como programador, não tenha de preocupar-se com isso.
- Os RDBMSs fornecem acessos aleatórios aos dados.
- Os RDBMSs têm sistemas predefinidos de privilégio.

Em termos mais concretos, utilizar um banco de dados relacional permite rápida e facilmente responder às consultas a respeito de onde estão os clientes, qual dos produtos está vendendo melhor ou que tipo de cliente gasta mais. Essas informações podem ajudar a melhorar o site para atrair e manter mais usuários. O banco de dados que utilizaremos nesta seção é o MySQL. Antes de entrarmos nas particularidades do MySQL no próximo capítulo, precisamos discutir:

- Conceitos e terminologia do banco de dados relational;
- Projeto de bancos de dados Web;
- Arquitetura do banco de dados Web.

Você aprenderá o seguinte, nesta parte do livro:

- O Capítulo 9 abrange a configuração básica necessária para conectar o banco de dados MySQL à Web. Você aprenderá como criar usuários, bancos de dados, tabelas e índices e os diferentes mecanismos de armazenamento de MySQL.
- O Capítulo 10 explica como consultar o banco de dados e adicionar, excluir e atualizar registros, tudo a partir da linha de comando.
- O Capítulo 11 explica como juntar o PHP e o MySQL de modo que você possa utilizar e administrar seu banco de dados a partir de uma interface Web. Você aprenderá dois métodos: utilizando a biblioteca MySQL do PHP e utilizando a camada de abstração de banco de dados PEAR:DB.

- O Capítulo 12 abrange a administração de MySQL em mais detalhes, incluindo sistema privilegiado, segurança e otimização.
- O Capítulo 13 abrange mecanismos de armazenamento em mais detalhes, incluindo transações, busca de texto completo e procedimentos armazenados.

Conceitos de banco de dados relacional

Os bancos de dados relacionais são, de longe, o tipo de banco de dados mais comumente utilizado. Eles dependem de uma boa base teórica em álgebra relacional. Você não precisa entender teoria relacional para utilizar um banco de dados relacional (o que é bom), mas precisa entender alguns conceitos básicos de banco de dados.

Tabelas

Os bancos de dados relacionais são compostos de relações, mais comumente chamadas de *tabelas*. Uma tabela é exatamente o que o nome sugere – uma tabela de dados. Se alguma vez utilizou uma planilha eletrônica, você já utilizou uma tabela relacional.

Vejamos um exemplo.

Na Figura 8.1, você pode ver uma tabela de exemplo. Ela contém os nomes e endereços dos clientes de uma livraria, Book-O-Rama.

CUSTOMERS

CustumerID	Nome	Endereço	Cidade
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 Norh Road	Yarraville

Figura 8.1 Os detalhes dos clientes de Book-O-Rama são armazenados em uma tabela.

A tabela tem um nome (Customers), várias colunas, cada uma correspondendo a um fragmento diferente de dados e linhas que correspondem a clientes individuais.

Colunas

Cada coluna na tabela tem um nome único e contém dados diferentes. Cada coluna tem um tipo de dados associado. Por exemplo, na tabela `Customers` na Figura 8.1, você pode ver que `CustomerID` é um inteiro e as outras três colunas são strings. As colunas às vezes são chamadas *campos* ou *atributos*.

Linhas

Cada linha na tabela representa um cliente diferente. Por causa do formato tabular, todas elas têm os mesmos atributos. As linhas também são chamadas de *registros* ou *tuplas*.

Valores

Cada linha consiste em um conjunto de valores individuais que correspondem às colunas. Cada valor deve ter o tipo de dados especificado pela sua coluna.

Chaves

Precisamos ter uma maneira de identificar cada cliente específico. Os nomes normalmente não são uma maneira muito boa de fazer isso – se você tiver um nome comum, provavelmente entenderá o

motivo. Considere Julie Smith da tabela Customers como exemplo. Se eu abrir minha lista telefônica, haverá listagens demais desse nome para contar.

Poderíamos distinguir Julie de várias maneiras. Provavelmente, ela seja a única Julie Smith vivendo no endereço dela. Falar de "Julie Smith, 25 Oak Street, Airport West" é muito incômodo e soa muito formal. Isso também exige utilizar mais de uma coluna na tabela.

O que fizemos nesse exemplo e o que você provavelmente vai fazer em suas aplicações, é atribuir um CustomerID único. Esse é o mesmo princípio que leva você a ter um número único de conta bancária ou número de associação. Isso facilita o armazenamento de seus detalhes em um banco de dados. Um número de identificação artificialmente atribuído pode ter a garantia de ser único. Poucas partes de informações reais, mesmo se utilizadas em combinação, têm essa propriedade.

A coluna de identificação em uma tabela é chamada de *chave* ou *chave primária*. Uma chave também pode consistir em várias colunas. Se por exemplo, tivéssemos escolhido referir a Julie como "Julie Smith, of 25 Oak Street, Airport West", a chave consistiria nas colunas Name, Address e City e não poderia ter garantia de ser única.

Normalmente, os bancos de dados consistem em várias tabelas e utilizam uma chave como uma referência de uma tabela para outra. Na Figura 8.2, adicionamos uma segunda tabela ao banco de dados. Esse banco de dados armazena pedidos feitos por clientes. Cada linha na tabela de pedidos representa uma única ordem, colocada por um único cliente. Sabemos quem o cliente é porque armazenamos seu CustomerID. Podemos ver a ordem com OrderID 2, por exemplo, e ver que o cliente com CustomerID 1 o colocou. Se você então vir a tabela Customers, pode ver que CustomerID 1 refere-se a Julie Smith.

CUSTOMERS

CustumerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 Norh Road	Yarraville

ORDERS

OrderID	CustumerID	Amount	Date
1	3	27,50	02-Abr-2000
2	1	12,99	15-Abr-2000
3	2	74,00	19-Abr-2000
4	3	6,99	01-Mai-2000

Figura 8.2 Cada ordem na tabela de pedidos refere-se a um cliente da tabela Customers.

O termo de banco de dados relacional para esse relacionamento é *chave estrangeira*. O CustomerID é a chave primária em Customers, mas quando ela aparece em outra tabela, como Orders, é referida como uma chave estrangeira.

Talvez você se pergunte por que escolhemos ter duas tabelas separadas – por que simplesmente não armazenamos o endereço da Julie na tabela Orders? Exploraremos isso em mais detalhe na próxima seção.

Esquemas

O conjunto completo dos designs de tabela para um banco de dados é chamado *esquema* de banco de dados. É similar a uma planta para o banco de dados. Um esquema deve mostrar as tabelas juntamente com suas colunas, os tipos de dados das colunas e indicar a chave primária de cada tabela e qualquer chave estrangeira. Um esquema não inclui dados, mas talvez você queira mostrar dados de

exemplo com seu esquema para explicar para que ele serve. O esquema pode ser mostrado como ele está nos diagramas que estamos utilizando, em *diagramas de relacionamento de entidade* (que não são abordados neste livro), ou em uma forma de texto, como:

```
Customers(CustomerID, Name, Address, City)
Orders(OrderID, , Amount, Date)
```

O termos sublinhados no esquema são chaves primárias na relação em que estão sublinhados. O termos em itálico são chaves estrangeiras na relação em que aparecem em itálico.

Relacionamentos

As chaves estrangeiras representam um relacionamento entre dados em duas tabelas. Por exemplo, o link desde `Orders` até `Customers` representa um relacionamento entre uma linha na tabela `Orders` e uma linha na tabela `Customers`.

Existem três tipos básicos de relacionamentos em um banco de dados relacional. Esses tipos são classificados de acordo com o número de itens em cada lado do relacionamento. Os relacionamentos podem ser de um para um, de um para muitos ou de muitos para muitos.

Um relacionamento de um para um significa que há um de cada item no relacionamento. Por exemplo, se colocamos endereços em uma tabela `Customers` separada, há um relacionamento de um para um entre eles. Você poderia ter uma chave estrangeira de `Addresses` para `Customer` ou ao contrário (as duas não são exigidas).

Em um relacionamento de um para muitos, uma linha em uma tabela é vinculada a muitas linhas na outra tabela. Nesse exemplo, um cliente talvez faça muitos pedidos. Nesses relacionamentos, a tabela que contém as muitas linhas terá uma chave estrangeira para a tabela com uma linha. Aqui, colocamos o `CustomerID` na tabela `Orders` para mostrar o relacionamento.

Em um relacionamento de muitos para muitos, muitas linhas em uma tabela são vinculadas a muitas linhas de outra tabela. Por exemplo, se tivéssemos duas tabelas, `Books` e `Authors`, você poderia achar que um livro foi escrito por dois co-autores, cada um dos quais escreveu outros livros, por conta própria ou possivelmente com outros autores. Esse tipo de relacionamento geralmente merece uma tabela inteira para si próprio, então talvez você tenha `Books`, `Authors` e `Books_Authors`. Essa terceira tabela só conteria as chaves das outras tabelas como chaves estrangeiras em pares, para mostrar quais autores estiveram envolvidos com quais livros.

Como projetar um banco de dados Web

Saber quando você precisa de uma nova tabela e que chave deve ser pode ser um tipo de arte. Você pode ler volumes enormes de informações sobre diagramas de relacionamento de entidade e normalização de banco de dados que estão além do escopo deste livro. Entretanto, a maior parte do tempo, você pode seguir alguns princípios básicos de design. Vamos considerar esses no contexto da Book-O-Rama.

Pense nos objetos do mundo real que está modelando

Quando cria um banco de dados, normalmente você modela itens do mundo real e relacionamentos, e armazena informações sobre esses objetos e relacionamentos.

Geralmente, cada classe de objetos do mundo real que você modela precisará de sua própria tabela. Pense sobre o assunto: queremos armazenar as mesmas informações sobre todos os nossos clientes. Se houver um conjunto de dados com a mesma "forma", podemos facilmente criar uma tabela correspondente para esses dados.

No exemplo da Book-O-Rama, queremos armazenar informações sobre os nossos clientes, os livros que vendemos e detalhes dos pedidos. Todos os clientes têm um nome e endereço. Os pedidos têm uma data, uma quantidade total e um conjunto de livros que foram pedidos. Os livros têm um ISBN, um autor, um título e um preço.

Isso sugere que precisamos pelo menos de três tabelas nesse banco de dados: Customers, Orders e Books. Esse esquema inicial é mostrado na Figura 8.3.

CUSTOMERS

CustomerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 North Road	Yarraville

ORDERS

OrderID	CustomerID	Amount	Date
1	3	27,50	02-Abr-2000
2	1	12,99	15-Abr-2000
3	2	74,00	19-Abr-2000
4	3	6,99	01-Mai-2000

BOOKS

ISBN	Author	Title	Price
0-672-31697-8	Michael Morgan	Java 2 for Professional Developers	34,99
0-672-31745-1	Thomas Down	Installing GNU/Linux	24,99
0-672-31509-2	Pruitt, et al.	Teach Yourself GIMP in 24 Hours	24,99

Figura 8.3 O esquema inicial consiste em Customers, Orders e Books.

Atualmente, não podemos dizer a partir do modelo quais livros foram encomendados em cada pedido. Trataremos disso em um minuto.

Evite armazenar dados redundantes

Anteriormente, perguntamos: "Por que simplesmente não armazenarmos o endereço da Julie Smith na tabela Orders?"

Se Julie encomendar da Book-O-Rama em várias ocasiões, o que esperamos que ela faça, acabaremos armazenando seus dados diversas vezes. Talvez você termine com uma tabela Orders parecida com a mostrada na Figura 8.4.

OrderID	Amount	Date	CustomerID	Name	Address	City
12	199,50	25-Abr-2000	1	Julie Smith	25 Oak Street	Airport West
13	43,00	29-Abr-2000	1	Julie Smith	25 Oak Street	Airport West
14	15,99	30-Abr-2000	1	Julie Smith	25 Oak Street	Airport West
15	23,75	01-Mai-2000	1	Julie Smith	25 Oak Street	Airport West

Figura 8.4 Um projeto de banco de dados que armazena dados redundantes ocupa espaço extra e pode causar anomalias nos dados.

Há dois problemas básicos com isso.

- O primeiro é que é um desperdício de espaço. Por que armazenar os detalhes de Julie três vezes se precisamos armazená-los somente uma vez?

- O segundo problema é que ele pode levar a *anomalias de atualização*, isto é, situações em que alteramos o banco de dados e terminamos com dados inconsistentes. A integridade dos dados é violada e não sabemos mais quais dados são corretos e quais são incorretos. Isso geralmente leva a perda de informações.

Três tipos de anomalias de atualização precisam ser evitados: anomalias de modificação, de inserção e de exclusão.

Se Julie mudar-se para uma nova casa enquanto tiver pedidos pendentes, precisaremos atualizar seu endereço em três lugares em vez de em apenas um, fazendo três vezes o trabalho. É fácil negligenciar esse fato e somente alterar seu endereço em um lugar, levando a dados inconsistentes no banco de dados (uma coisa muito ruim). Esses problemas são chamados *anomalias de modificação* porque ocorrem quando estamos tentando modificar o banco de dados.

Com esse projeto, precisamos inserir os detalhes da Julie toda vez que recebemos um pedido, então sempre devemos verificar e nos certificar de que os seus detalhes são consistentes com as linhas existentes na tabela. Se não verificamos, podemos terminar com duas linhas de informações contraditórias sobre Julie. Por exemplo, uma linha talvez informe que Julie vive em Airport West e outra talvez informe que ela vive em Airport. Isso é chamado uma *anomalia de inserção* porque ocorre quando os dados estão sendo inseridos.

O terceiro tipo de anomalia é chamado uma *anomalia de exclusão* porque ocorre (adivinhe) quando estamos excluindo linhas do banco de dados. Por exemplo, imagine que quando um pedido foi despachado, o excluímos do banco de dados. Quando todos os pedidos atuais da Julie forem preenchidos, todos eles são excluídos das tabelas de pedidos. Isso significa que não temos mais um registro do endereço da Julie. Não podemos enviar para ela nenhuma oferta especial e da próxima vez que ela quiser encomendar algo de nós, teremos de obter todos os seus detalhes novamente.

Geralmente você deve projetar o banco de dados para que nenhuma dessas anomalias ocorra.

Utilize valores atômicos de coluna

Isso significa que em cada atributo de cada linha, armazenamos somente uma informação. Por exemplo, precisamos saber quais os livros que compõem cada pedido. Há várias maneiras de fazer isso.

Poderíamos adicionar uma coluna à tabela `Orders` listando todos os livros que foram encomendados, como mostrado na Figura 8.5.

ORDERS

OrderID	CustomerID	Amount	Date	Books Ordered
1	3	27,50	02-Abr-2000	0-672-31697-8
2	1	12,99	15-Abr-2000	0-672-31745-1, 0-672-31509-2
3	2	74,00	19-Abr-2000	0-672-31697-8
4	3	6,99	01-Mai-2000	0-672-31745-1, 0-672-31509-2, 0-672-31697-8

Figura 8.5 Com esse projeto, o atributo `Books Ordered` em cada linha tem diversos valores.

Essa não é uma boa idéia por algumas razões. O que realmente estamos fazendo é aninhar uma tabela inteira dentro de uma coluna – uma tabela que relaciona pedidos de livros. Quando você faz isso dessa maneira, fica mais difícil responder a perguntas como "Quantas cópias de *Java 2 for Professional Developers* foram encomendadas?" O sistema simplesmente não pode mais contar os campos correspondentes. Em vez disso, ele tem de analisar sintaticamente cada valor de atributo para ver se contém uma correspondência em algum lugar dentro dele.

Como na realidade estamos criando uma tabela dentro de uma tabela, de fato precisamos apenas criar essa nova tabela, chamada `Order_Items` e mostrada na Figura 8.6.

ORDERS_ITEMS

OrderID	ISBN	Quantity
1	0-672-31697-8	1
2	0-672-31745-1	2
2	0-672-31509-2	1
3	0-672-31509-2	1
4	0-672-31745-1	1
4	0-672-31509-2	2
4	0-672-31509-2	1

Figura 8.6 Esse projeto torna mais fácil procurar determinados livros que foram encomendados.

Essa tabela fornece um link entre a tabela `Orders` e a tabela `Books`. Esse tipo de tabela é comum quando há um relacionamento de muitos para muitos entre dois objetos – nesse caso, um pedido talvez consista em muitos livros e cada livro pode ser encomendo por muitas pessoas.

Escolha chaves sensíveis

Certifique-se de que as chaves que você escolhe sejam únicas. Nesse caso, criamos uma chave especial para clientes (`CustomerID`) e para pedidos (`OrderID`) porque esses objetos do mundo real talvez não tenham naturalmente um identificador que garanta que eles sejam únicos. Não precisamos criar um identificador único para livros – isso já foi feito, na forma do ISBN. Para `Order_Item`, você pode adicionar uma chave extra se quiser, mas a combinação dos dois atributos `OrderID` e `ISBN` será única contanto que mais de uma cópia do mesmo livro em um pedido seja tratada como uma linha. Por essa razão, a tabela `Order_Items` tem uma coluna `Quantity`.

Pense nas perguntas que deseja fazer ao banco de dados

Continuando da última seção, pense nas perguntas que você gostaria que o banco de dados respondesse. (Pense outra vez naquelas perguntas que mencionamos no início do capítulo. Por exemplo, quais são os livros best-seller da Book-O-Rama?) Certifique-se de que o banco de dados contém todos os dados exigidos e que existem os links apropriados entre tabelas para responder às suas perguntas.

Evite projetos com muitos atributos vazios

Se quiséssemos adicionar resenhas de livro ao banco de dados, há pelo menos duas maneiras como poderíamos fazer isso. Essas duas abordagens são mostradas na Figura 8.7.

BOOKS

ISBN	Author	Title	Price	Review
0-672-31687-8	Michael Morgan	Java 2 for Professional Developers	34.99	
0-672-31745-1	Thomas Down	Installing Debian GNU/Linux	24.99	
0-672-31509-2	Pruitt, et al.	Teach Yourself GIMP in 24 Hours	24.99	

BOOK_REVIEWS

ISBN	Review

Figura 8.7 Para adicionar resenhas, podemos adicionar uma coluna Reviews à tabela `Books` ou adicionar uma tabela especificamente para resenhas.

A primeira maneira significa adicionar uma coluna `Review` à tabela `Books`. Dessa maneira, há um campo para a resenha ser adicionada a cada livro. Se muitos livros estiverem no banco de dados e a pessoa que faz a resenha não planeja resenhar todos eles, muitas linhas não terão um valor nesse atributo. Isso é chamado *ter um valor nulo*.

Ter muitos valores nulos no banco de dados é uma péssima idéia. Desperdiça espaço de armazenamento e causa problemas ao elaborar totais e outras funções em colunas numéricas. Quando um usuário vê um nulo em uma tabela, ele não sabe se é porque esse atributo é irrelevante, se há um erro no banco de dados ou se os dados simplesmente não foram inseridos ainda.

Geralmente você pode evitar problemas com muitos nulos utilizando um projeto alternativo. Nesse caso, podemos utilizar o segundo projeto proposto na Figura 8.7. Aqui, somente os livros com uma resenha são listados na tabela `Book_Reviews`, juntamente com sua resenha.

Observe que esse projeto está baseado na idéia de ter um único resenhador interno. Poderíamos igualmente deixar simplesmente aos clientes as resenhas do autor. Se quiséssemos fazer isso, poderíamos adicionar o `CustomerID` à tabela `Book_Reviews`. Se você tem muitas resenhas por livro, deve introduzir um único identificador para cada.

Resumo dos tipos de tabela

Você normalmente descobrirá que seu projeto de banco de dados terminará consistindo em dois tipos de tabela:

- Tabelas simples que descrevem um objeto do mundo real. Esses talvez também contenham chaves para outros objetos simples onde há um relacionamento de um para um ou um para muitos. Por exemplo, um cliente talvez tenha muitos pedidos, mas um pedido é feito por um único cliente. Portanto, colocamos uma referência para o cliente no pedido.
- Tabelas de vinculação que descrevem um relacionamento de muitos para muitos entre dois objetos reais como o relacionamento entre `Orders` e `Books`. Essas tabelas freqüentemente estão associadas a algum tipo de transação do mundo real.

Arquitetura de banco de dados Web

Agora que discutimos a arquitetura interna do banco de dados, examinaremos a arquitetura externa de um sistema de banco de dados Web e discutiremos a metodologia para desenvolver um sistema de banco de dados Web.

Arquitetura

A operação básica de um servidor Web é mostrada na Figura 8.8. Esse sistema consiste em dois objetos: um navegador Web e um servidor Web. É necessário um link de comunicação entre eles. Um navegador Web faz uma solicitação ao servidor. O servidor envia de volta uma resposta. Essa arquitetura atende bem a um servidor que fornece páginas estáticas. A arquitetura que fornece um Web site baseado em um banco de dados é um pouco mais complexa.

Figura 8.8 O relacionamento cliente/servidor entre um navegador Web e um servidor Web exige comunicação.

As aplicações de banco de dados Web que construiremos neste livro seguem uma estrutura do banco de dados Web geral que é mostrada na Figura 8.9. A maior parte dessa estrutura já deve ser familiar para você.

Uma típica transação de banco de dados Web consiste nas seguintes etapas, que são numeradas na Figura 8.9. Examinaremos as etapas no contexto do exemplo Book-O-Rama.

Figura 8.9 A arquitetura básica de banco de dados Web consiste no navegador Web, no servidor Web, no mecanismo de criação de script e no servidor de bancos de dados.

1. Um navegador Web do usuário emite uma solicitação de HTTP para uma página Web particular. Por exemplo, talvez ele tenha solicitado uma pesquisa de todos os livros da Book-O-Rama escritos por Laura Thomson, utilizando um formulário de HTML. A página de resultados de pesquisa é chamada `results.php`.
2. O servidor Web recebe a solicitação para `results.php`, recupera o arquivo e passa para o mecanismo de PHP para processamento.
3. O mecanismo de PHP começa a analisar sintaticamente o script. Dentro do script está um comando para conectar-se ao banco de dados e executar uma consulta (realizar a pesquisa dos livros). O PHP abre uma conexão com o servidor do MySQL e envia a consulta apropriada.
4. O servidor do MySQL recebe a consulta de banco de dados e a processa e envia os resultados – uma lista de livros – de volta para o mecanismo de PHP.
5. O mecanismo de PHP termina a execução do script, que normalmente envolverá a formatação dos resultados da consulta bem em HTML. Então, ele retorna a HTML resultante para o servidor Web.
6. O servidor Web passa a HTML de volta para o navegador, onde o usuário pode ver a lista de livros que solicitou.

O processo é basicamente o mesmo independente de quais mecanismos de criação de script ou servidor de banco de dados você utiliza. Freqüentemente, o software de servidor Web, o mecanismo de PHP e o servidor do banco de dados executam na mesma máquina. Entretanto, também é bastante comum o servidor do banco de dados executar em uma máquina diferente. Talvez você faça isso por razões de segurança, aumento de capacidade ou distribuição de carga. Da perspectiva do desenvolvimento, esse será quase o mesmo trabalho, mas talvez ofereça algumas vantagens significativas no desempenho.

Leitura adicional

Neste capítulo, abrangemos algumas diretrizes para projeto de banco de dados relacional. Se quiser aprofundar-se na teoria por trás dos bancos de dados relacionais, você pode tentar ler livros de alguns gurus relacionais como C.J. Date. Mas, tenha cuidado, que o material pode ficar muito teórico e talvez não seja imediatamente relevante para um desenvolvedor comercial Web. O banco de dados Web médio tende a não ser tão complicado.

A seguir

No próximo capítulo, começaremos a configurar o banco de dados do MySQL. Primeiro, você aprenderá a configurar um banco de dados do MySQL para a Web, consultá-lo e então consultá-lo a partir do PHP.

9

Criando bancos de dados Web

NESTE CAPÍTULO DISCUTIREMOS SOBRE como configurar um banco de dados do MySQL para utilização em um Web site.

Os tópicos-chave incluídos neste capítulo são:

- Criação de um banco de dados;
- Configuração de usuários e privilégios;
- Introdução aos privilégios de sistema;
- Criação de tabelas de banco de dados;
- Criação de índices;
- Escolhendo tipos de colunas em MySQL.

Neste capítulo, vamos prosseguir com a aplicação on-line da livraria Book-O-Rama discutida no último capítulo. Como um lembrete, eis o esquema para a aplicação Book-O-Rama:

```
Customers(CustomerID, Name, Address, City)

Orders(OrderID, CustomerID, Amount, Date)

Books(ISBN, Author, Title, Price)

Order_Items(OrderID, ISBN, Quantity)

Book_Reviews(ISBN, Reviews)
```

Lembre-se de que as chaves primárias são sublinhadas e chaves estrangeiras têm um sublinhado pontilhado.

A fim de utilizar o material nesta seção, você deve ter acesso ao MySQL. Em geral, isso significa que você:

- Completou a instalação básica do MySQL no seu servidor Web. Isso inclui:
    - Instalar os arquivos;
    - Configurar um usuário para MySQL a fim de executar como tal;
    - Configurar seu caminho;
    - Executar `mysql_install_db`, se exigido;
    - Configurar a senha root;
    - Excluir o usuário anônimo e o banco de dados de teste;

- Iniciar o servidor do MySQL e configurá-lo para executar automaticamente.

  Se você completou todas essas tarefas, pode seguir direto e ler este capítulo. Se ainda não completou, pode encontrar instruções sobre como fazer isso no Apêndice A.

  Se você tiver problemas em algum ponto deste capítulo, talvez seja porque seu sistema do MySQL não esteja configurado corretamente. Se isso acontecer, consulte outra vez esta lista e o Apêndice A para certificar-se de que sua configuração está correta.

- Tem acesso ao MySQL em uma máquina que você não administra como um serviço de hospedagem na Web, uma máquina no local de trabalho e assim por diante.

  Se esse for o caso, para lidar com os exemplos ou criar seu próprio banco de dados, você precisará fazer seu administrador configurar um usuário e banco de dados para você trabalhar e informar o nome de usuário, a senha e o nome de banco de dados atribuídos a você.

  Você pode pular as seções deste capítulo que explicam como configurar usuários e bancos de dados ou lê-los para explicar melhor o que você precisa para seu administrador de sistema. Como um usuário normal, você não será capaz de executar os comandos para criar usuários e bancos de dados.

Os exemplos neste capítulo foram todos construídos e testados com a mais recente versão do MySQL 5.0, na época em que o livro estava sendo escrito. Algumas versões anteriores do MySQL têm menos funcionalidades. Você deve instalar ou atualizar a versão mais atual no momento em que estiver lendo este livro. Você pode fazer download da versão atual no site do MySQL em http://mysql.com.

Usando o MySQL Monitor

Você notará que os exemplos do MySQL neste capítulo e no próximo terminam cada comando com um ponto-e-vírgula (;). Isso informa ao MySQL para executar o comando. Se você deixar o ponto-e-vírgula fora, não acontecerá nada. Esse é um problema comum para os usuários iniciantes.

Isso também significa que você pode ter novas linhas no meio de um comando. Utilizamos isso para facilitar ainda mais a leitura dos exemplos. Você verá onde fizemos isso porque o MySQL fornece um símbolo de continuação. Trata-se de uma seta parecida com:

```
mysql> grant select
    ->
```

Isso significa que o MySQL está esperando mais entrada. Até digitar o ponto-e-vírgula, você obterá esses caracteres toda vez que pressionar Enter.

Outro ponto a ser observado é que instruções de SQL não fazem distinção entre letras maiúsculas e minúsculas, mas os banco de dados e os nomes de tabela podem fazer – veremos melhor isso mais tarde.

Como efetuar logon no MySQL

Para fazer logon, vá para uma interface de linha de comando na sua máquina e digite o seguinte:

```
mysql -h hostname -u username -p
```

O comando `mysql` invoca o monitor do MySQL. Esse é um cliente de linha de comando que o conecta ao servidor do MySQL.

A opção `-h` é utilizada para especificar o host ao qual você quer conectar-se; isto é, a máquina em que o servidor do MySQL está executando. Se estiver executando esse comando na mesma máquina que o servidor do MySQL, você pode omitir essa opção e o parâmetro *nome_do_host*. Se não estiver, você deve substituir o parâmetro *nome_do_host* pelo nome da máquina em que o servidor do MySQL está executando.

A opção -u é utilizada para especificar o *nome_do_usuário* com o qual você quer conectar-se. Se não especificar, o padrão será o nome de usuário com o qual você efetuou logon no sistema operacional.

Se você instalou o MySQL na sua própria máquina ou servidor, precisará efetuar logon como root e criar o banco de dados que utilizaremos nesta seção. Assumindo que você tem uma instalação inteligente, root é o único usuário com que você terá de iniciar. Se você estiver utilizando o MySQL em uma máquina administrada por mais alguém, utilize o nome de usuário que eles lhe deram.

A opção -p instrui o servidor que você quer conectar-se utilizando uma senha. Você pode deixá-la de lado se uma senha não foi configurada para o usuário com o qual você está efetuando logon.

Se estiver efetuando logon como root e não configurou uma senha para root, recomendo que você consulte o Apêndice A e faça isso agora mesmo. Sem essa senha, seu sistema é inseguro.

Você não precisa incluir a senha nessa linha. O servidor do MySQL a solicitará. De fato, é melhor se você não incluir. Se você inserir a senha na linha de comando, ela aparecerá como texto simples na tela e será bem simples para outros usuários descobrirem.

Depois que inseriu o comando anterior, você deve obter uma resposta assim:

```
Enter password: ****
```

(Se essa não funcionou, verifique se o servidor do MySQL está executando e se o comando mysql está em algum lugar no seu caminho.)

Você deve inserir sua senha. Se tudo der certo, deve aparecer uma resposta assim:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.52-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Na sua própria máquina, se não obtiver uma resposta semelhante a essa, certifique-se de que executou mysql_install_db se exigido, configurou a senha root e a digitou corretamente. Se não for sua máquina, certifique-se de que digitou a senha corretamente.

Agora você deve estar em um prompt de comando do MySQL, pronto para criar o banco de dados. Se você estiver utilizando sua própria máquina, siga as diretrizes na próxima seção. Se estiver utilizando a máquina de outra pessoa, isso já deve ter sido feito para você. Você pode ir diretamente para a seção "Utilizando o banco de dados certo". Você talvez queira ler as seções relacionadas à visão geral, mas não poderá executar os comandos especificados. (Ou pelo menos não deveria poder!)

Criando bancos de dados e usuários

O sistema de banco de dados do MySQL pode suportar muitos bancos de dados diferentes. Geralmente, você terá um banco de dados por aplicação. No exemplo Book-o-Rama, o banco de dados será chamado books.

Criando o banco de dados

Esta é a parte mais fácil. No prompt de comando do MySQL, digite:

```
mysql> create database nomedodb;
```

Você deve substituir *nomedodb* pelo nome do banco de dados que deseja criar. Para começar a criar o exemplo Book-o-Rama, você pode criar um banco de dados chamado books.

É isso aí. Você deve ver uma resposta assim:

```
Query OK, 1 row affected (0.06 sec)
```

Isso significa que tudo funcionou. Se você não obtiver essa resposta, certifique-se de que digitou o ponto-e-vírgula no final da linha. Um ponto-e-vírgula diz ao MySQL que você concluiu e realmente deve executar o comando.

Definindo usuários e privilégios

Um sistema do MySQL pode ter muitos usuários. O usuário root geralmente deve ser utilizado somente para propósitos de administração, por razões de segurança. Para cada usuário que precisar utilizar o sistema, você precisará configurar uma conta e senha. Esses não precisam ser os mesmos nomes de usuário e senhas fora do MySQL (por exemplo, nomes de usuário e senhas UNIX ou NT). O mesmo princípio se aplica a root. É uma boa idéia ter senhas diferentes para o sistema e MySQL, especialmente quando se trata da senha root.

Não é obrigatório configurar senhas para usuários, mas recomendamos configurar senhas para todos os usuários que forem criados. Para os propósitos de configurar um banco de dados Web, é uma boa idéia configurar pelo menos um usuário por aplicação Web. Você talvez pergunte, "Por que faria isso?" – a resposta está nos privilégios.

Introdução ao sistema de privilégios do MySQL

Um dos melhores recursos do MySQL é que ele suporta um sistema sofisticado de privilégio. Um *privilégio* é o direito de realizar uma ação particular em um objeto particular e está associado a um usuário particular. O conceito é muito semelhante ao das permissões de arquivo. Quando você cria um usuário dentro do MySQL, concede a ele um conjunto de privilégios para especificar o que ele pode e e o que não pode fazer dentro do sistema.

Princípio do menor privilégio

O princípio do menor privilégio pode ser utilizado para melhorar a segurança de qualquer sistema de computador. Esse é um princípio básico, mas muito importante que é freqüentemente negligenciado. O princípio é o seguinte:

> Um usuário (ou processo) deve ter o nível mais baixo de privilégio exigido a fim de realizar a tarefa atribuída a ele.

Ele se aplica no MySQL assim como se aplica em qualquer outra parte. Por exemplo, para executar consultas Web, um usuário não precisa de todos os privilégios a que root tem acesso. Portanto, devemos criar outro usuário que tenha apenas os privilégios necessários para acessar o banco de dados que acabamos de criar.

Configurando usuários: o comando GRANT

Os comandos GRANT e REVOKE são utilizados para fornecer e retirar direitos dos usuários do MySQL em quatro níveis de privilégio. Esses níveis são:

- Global;
- Banco de dados;
- Tabela;
- Coluna.

Veremos em um momento como cada um desses pode ser aplicado.

O comando GRANT é utilizado para criar usuários e fornecer privilégios. A forma geral do comando GRANT é:

```
GRANT privilégios [colunas]
ON item
TO nome_do_usuário [IDENTIFIED BY 'senha']
[REQUIRE ssl_opções]
[WITH [GRANT OPTION | opções_limite
```

As cláusulas entre colchetes são opcionais. Há vários marcadores de lugar nessa sintaxe. O primeiro, *privilégios*, dever ser uma lista de privilégios separados por vírgula. O MySQL tem um conjunto definido desses. Eles são descritos na próxima seção.

O marcador de lugar `colunas` é opcional. Você pode utilizar para especificar privilégios individualmente por coluna. Você pode utilizar um único nome de coluna ou uma lista de coluna de nomes separados por vírgulas.

O marcador de lugar `item` é o banco de dados ou tabela a que os novos privilégios se aplicam. Você pode conceder privilégios em todos os bancos de dados especificando `*.*` como o `item`. Isso é chamado de concessão de privilégios *globais*. Você também pode fazer isso especificando unicamente * se não estiver utilizando nenhum banco de dados particular. Mais comumente, você especificará todas as tabelas em um banco de dados como `nomedobd.*`, em uma única tabela como `nomedobd.nomedatabela` ou em colunas específicas especificando `nomedobd.nomedatabela` e algumas colunas específicas no marcador de lugar `colunas`. Esses representam os três outros níveis de privilégio disponíveis: `banco de dados`, `tabela` e `coluna`, respectivamente. Se você estiver utilizando um banco de dados específico ao emitir esse comando, `nomedatabela` por si só será interpretado como uma tabela no banco de dados atual.

O `nome_do_usuário` deve ser o nome com o qual você deseja que o usuário efetue logon no MySQL. Lembre-se de que ele não tem de ser o mesmo que o nome de login de sistema. O `nome_do_usuário` no MySQL também pode conter um nome de host. Você pode utilizar isso para diferenciar entre, digamos, `laura` (interpretado como `laura@localhost`) e `laura@somewhere.com`. Isso é bem útil visto que usuários de domínios diferentes freqüentemente têm o mesmo nome. Isso também aumenta a segurança porque permite especificar a partir de onde os usuários podem se conectar e até quais tabelas ou bancos de dados eles podem acessar a partir de uma localização particular.

A `senha` deve ser a senha com a qual você deseja que o usuário efetue o login. Aplicam-se as regras normais para selecionar senhas. Discutiremos segurança em detalhes mais tarde, mas uma senha não deve ser facilmente adivinhável. Isso significa que uma senha não deve ser uma palavra de dicionário nem mesmo o nome de usuário. Idealmente, ela conterá uma mistura de caracteres não-alfabéticos em letras minúsculas e maiúsculas.

A cláusula `REQUIRE` permite especificar que o usuário deve se conectar via Secure Sockets Layer (SSL) e especificar outras opções SSL. Para obter mais informações sobre conexões SSL ao MySQL, consulte o manual do MySQL.

A opção `WITH GRANT OPTION`, se especificada, permite que o usuário especificado conceda seus próprios privilégios a outros.

Em vez disso, você pode especicar a cláusula `WITH` assim

```
MAX_QUERIES_PER_HOUR n
```

ou

```
MAX_UPDATES_PER_HOUR n
```

ou

```
MAX_CONNECTIONS_PER_HOUR n
```

Essas cláusulas permitem limitar o número de consultas, atualizações ou conexões por hora que um usuário pode fazer. Elas são úteis para limitar a carga individual de um usuário em sistemas compartilhados.

Os privilégios são armazenados em quatro tabelas de sistema, no banco de dados chamado `mysql`. Essas quatro tabelas chamam-se `mysql.user`, `mysql.db`, `mysql.tables_priv` e `mysql.columns_priv`, e se relacionam diretamente com os quatro níveis de privilégio mencionados antes. Como uma alternativa à `GRANT`, você pode alterar essas tabelas diretamente. Discutiremos isso em mais detalhes no Capítulo 12.

Tipos e níveis de privilégio

Existem três tipos de privilégios básicos no MySQL: privilégios adequados para conceder a usuários regulares, privilégios adequados para administradores e dois privilégios especiais. Qualquer usuário pode ser receber qualquer um desses privilégios, mas normalmente é sensato restringir os do tipo administrador para administradores, de acordo com o princípio do menor privilégio.

Você deve conceder aos usuários somente os privilégios sobre os bancos de dados e tabelas que eles precisam utilizar. Você não deve conceder acesso ao banco de dados `mysql` a qualquer pessoa exceto a um administrador. Esse é o lugar em que todos os usuários, senhas etc. são armazenados. (Veremos esse banco de dados no Capítulo 12.)

Os privilégios para usuários regulares se relacionam diretamente aos tipos de comandos específicos de SQL e se um usuário tem ou não permissão de executá-los. Discutiremos esses comandos de SQL em detalhes no próximo capítulo. Por enquanto, fornecemos uma descrição conceitual do que eles fazem. Esses privilégios são mostrados na Tabela 9.1. Os itens sob a coluna intitulada "Aplica-se a" listam os objetos aos quais privilégios desse tipo podem ser concedidos.

Tabela 9.1 **Privilégios para usuários**

Privilégio	Aplica-se a	Descrição
`SELECT`	tabelas, colunas	Permite que os usuários selecionem linhas (registros) de tabelas.
`INSERT`	tabelas, colunas	Permite que os usuários insiram novas linhas'em tabelas.
`UPDATE`	tabelas, colunas	Permite que os usuários modifiquem valores em linhas existentes de tabela.
`DELETE`	tabelas	Permite que os usuários excluam linhas existentes de tabela.
`INDEX`	tabelas	Permite que os usuários criem e excluam índices em tabelas particulares.
`ALTER`	tabelas	Permite que os usuários alterem a estrutura de tabelas existentes, por exemplo, adicionar colunas, renomear colunas ou tabelas e alterar tipos de dados de colunas.
`CREATE`	bancos de dados, tabelas	Permite que os usuários criem novos bancos de dados ou tabelas. Se um particular banco de dados ou tabela for especificado na instrução `GRANT`, você pode somente criar (`CREATE`) esse banco de dados ou tabela, o que significa que eles terão de ser excluídos (`DROP`) primeiro.
`DROP`	bancos de dados, tabelas	Permite que os usuários excluam (delete) bancos de dados ou tabelas.

A maioria dos privilégios para usuários regulares é relativamente inofensiva em termos de segurança do sistema. O privilégio `ALTER` pode ser utilizado para contornar o sistema de privilégios mediante o renomeação das tabelas, mas a maioria dos usuários precisa deles. A segurança representa sempre uma relação de troca entre usabilidade e segurança. Você deve tomar sua própria decisão quando se trata de `ALTER`, mas freqüentemente ele é concedido aos usuários.

Além dos privilégios listados na Tabela 9.1, existe um privilégio `REFERENCES` que atualmente não é utilizado; e existe um privilégio `GRANT` que é concedido com `WITH GRANT OPTION` em vez de na lista *privilégios*.

A Tabela 9.2 mostra os privilégios adequados para utilização pelos usuários administrativos.

Tabela 9.2 **Privilégios para administradores**

Privilégio	Descrição
CREATE TEMPORARY TABLES	Permite que o administrador utilize a palavra-chave TEMPORARY em uma instrução CREATE TABLE.
FILE	Permite que os dados sejam lidos e transferidos para as tabelas a partir de arquivos e vice-versa.
LOCK TABLES	Permite o uso explícito de uma instrução LOCK TABLES.
PROCESS	Permite que o administrador visualize processos de servidor que pertençam a todos os usuários.
RELOAD	Permite que o administrador recarregue tabelas e revogue privilégios, hosts, logs e tabelas.
REPLICATION CLIENT	Permite o uso de SHOW STATUS em masters e slaves de replicação. A replicação é explicada no Capítulo 12.
REPLICATION SLAVE	Permite que os servidores slaves de replicação se conectem ao servidor master. A replicação é explicada no Capítulo 12.
SHOW DATABASES	Permite que uma lista de todos os bancos de dados seja vista com a instrução SHOW DATABASES. Sem esse privilégio, os usuários vêem apenas os bancos dados aos quais têm outros privilégios.
SHUTDOWN	Permite que um administrador desligue o servidor MySQL.
SUPER	Permite que um administrador mate os threads que pertençam a qualquer usuário.

É possível conceder esses privilégios para não-administradores, mas você deve ser extremamente cauteloso se estiver pensando em fazer isso.

O privilégio FILE é um pouco diferente. Esse privilégio é útil aos usuários porque carregar dados a partir de arquivos pode economizar muito tempo de reinserção repetitiva de dados para armazená-los no banco de dados. Entretanto, o carregamento de arquivo pode ser utilizado para carregar quaisquer arquivos que o servidor do MySQL possa ver, incluindo bancos de dados pertencentes a outros usuários e, potencialmente, arquivos de senha. Conceda-o com cautela ou ofereça-se para carregar os dados para o usuário.

Existem também dois privilégios especiais e esses são mostrados na Tabela 9.3.

Tabela 9.3 **Privilégios especiais**

Privilégio	Descrição
ALL	Concede todos os privilégios listados nas Tabelas 9.1 e 9.2. Você também pode escrever ALL PRIVILEGES em vez de ALL.
USAGE	Não concede nenhum privilégio. Isso criará um usuário e o permitirá efetuar logon, mas não permitirá fazer nada. Normalmente, você prosseguirá para adicionar mais privilégios mais tarde.

O comando REVOKE

O oposto de GRANT é REVOKE. Esse comando é utilizado para revogar os privilégios de um usuário. Sua sintaxe é muito semelhante a de GRANT:

```
REVOKE privilégios [(colunas)]
ON item
FROM user_name
```

Se concedeu a cláusula WITH GRANT OPTION, você pode revogar fazendo:

```
REVOKE GRANT OPTION
ON item
FROM nome_do_usuário
```

Exemplos utilizando GRANT e REVOKE

Para configurar um administrador, você pode digitar:

```
mysql> grant all, grant
    -> on *
    -> to fred identified by 'mnb123'
    -> with grant option;
```

Isso concede todos os privilégios sobre todos os bancos de dados a um usuário chamado Fred com a senha mnb123 e permite a ele repassar esses privilégios.

Provavelmente você não quer esse usuário no sistema, então vá em frente e o revogue:

```
mysql> revoke all
    -> from fred;
```

Agora vamos configurar um usuário regular sem privilégios:

```
mysql> grant usage
    -> on books.*
    -> to sally identified by 'magic123';
```

Depois de conversar com Sally, sabemos um pouco mais sobre o que ela quer fazer, então podemos fornecer-lhe os privilégios apropriados:

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on books.*
    -> to sally;
```

Observe que não precisamos especificar a senha da Sally para fazer isso.

Se decidirmos que Sally é capaz de fazer algo no banco de dados, talvez decidamos reduzir seus privilégios:

```
mysql> revoke alter, create, drop
    -> on books.*
    -> from sally;
```

E mais tarde, quando ela não precisar mais utilizar o banco de dados, podemos revogar seus privilégios de uma só vez:

```
mysql> revoke all
    -> on books.*
    -> from sally;
```

Configurando um usuário para a Web

Você precisará configurar um usuário para seus scripts de PHP a fim de conectar-se ao MySQL. Novamente podemos aplicar o princípio de menor privilégio: o que os scripts devem ser capazes de fazer?

Na maioria dos casos, eles precisarão somente SELECT, INSERT, DELETE e UPDATE (selecionar, inserir, excluir e atualizar) as linhas a partir de tabelas. Você pode configurar isso da seguinte maneira:

```
mysql> grant select, insert, delete, update
    -> on books.*
    -> to bookorama identified by 'bookorama123';
```

Obviamente, por razões de segurança, você deve escolher uma senha melhor do que essa.

Se utilizar um serviço de hospedagem na Web, você normalmente obterá acesso a outros privilégios do tipo privilégio de usuário em um banco de dados que o serviço de hospedagem cria para você. O serviço de hospedagem em geral fornecerá o mesmo *nome_do_usuário* e *senha* para utilização de linha de comando (configurar tabelas e assim por diante) e para conexões de script Web (consultar o banco de dados). Isso é marginalmente menos seguro. Você pode configurar um usuário com esse nível de privilégio da seguinte maneira:

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on books.*
    -> to bookorama identified by 'bookorama123';
```

Prossiga e configure essa segunda versão do usuário porque isso é o que precisaremos utilizar na próxima seção.

Efetuando o logout como root

É possível efetuar logout do monitor MySQL digitando `quit`. Faça logon novamente como usuário Web para testar se tudo está funcionando corretamente. Se a instrução `GRANT` tiver sido executada, mas você teve o acesso negado ao tentar fazer logon significa que não excluiu os usuários anônimos como parte do processo de instalação. Faça logon novamente como root e consulte o Apêndice A para obter instruções de como excluir contas anônimas. Só então você deve conseguir fazer logon como usuário Web.

Utilizando o banco de dados certo

Se alcançou esta etapa, você deve ter efetuado logon com uma conta de nível do usuário do MySQL pronta para testar o código de exemplo, ou porque acabou de configurá-lo ou porque seu administrador de servidor Web o configurou para você.

A primeira ação necessária ao efetuar logon é especificar qual banco de dados quer utilizar. Você pode fazer isso digitando:

```
mysql> use nomedodb;
```

onde *nomedodb* é o nome de seu banco de dados.

Alternativamente, você pode evitar o comando `use` especificando o banco de dados quando efetuar logon, da seguinte maneira:

```
mysql -D nomedodb -h nomedohost -u nomedousuário -p
```

Nesse exemplo, utilizaremos o banco de dados de livros:

```
mysql> use books;
```

Quando você digitar esse comando, o MySQL deve fornecer uma resposta como:

```
Database changed
```

Se você não seleciona um banco de dados antes de começar a trabalhar, o MySQL fornecerá uma mensagem de erro como:

```
ERROR 1046: No Database Selected
```

Criando tabelas de banco de dados

O próximo passo na configuração do banco de dados é realmente criar as tabelas. Você pode fazer isso utilizando o comando de SQL `CREATE TABLE`. A forma geral de uma instrução `CREATETABLE` é:

```
CREATE TABLE nomedatabela(colunas)
```

Você deve substituir o marcador de lugar *nomedatabela* pelo nome da tabela que deseja criar e o marcador de lugar *colunas* por uma lista de nomes de colunas separados por vírgulas da sua tabela. Cada coluna terá um nome seguido por um tipo de dados.

Eis o esquema de Book-O-Rama:

Customers(<u>CustomerID</u>, Name, Address, City)

Orders(<u>OrderID</u>, *CustomerID*, Amount, Date)

Books(<u>ISBN</u>, Author, Title, Price)

Order_Items(*<u>OrderID</u>*, *<u>ISBN</u>*, Quantity)

Book_Reviews(*<u>ISBN</u>*, Review)

A Listagem 9.1 mostra o SQL para criar essas tabelas, assumindo que você já criou o banco de dados chamado `books`. Você pode localizar essa SQL no CD-ROM no arquivo `chapter9/bookorama.sql`.

Você pode executar um arquivo existente de SQL, como por exemplo, um carregado a partir do CD-ROM, usando MySQL digitando:

```
> mysql -h host -u bookorama -D books -p < bookorama.sql
```

(Lembre-se de substituir *host* pelo nome do seu host.)

Utilizar a redireção de arquivo é muito útil para isso porque significa que você pode editar seu SQL no editor de textos de sua escolha antes de executá-lo.

Listagem 9.1 **bookorama.sql – O SQL para criar as tabelas para a Book-O-Rama**

```
create table customers
( customerid int unsigned not null auto_increment primary key,
  name char(30) not null,
  address char(40) not null,
  city char(20) not null
);

create table orders
( orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null,
amount float(6,2),
  date date not null
);

create table books
(  isbn char(13) not null primary key,
   author char(30),
   title char(60),
   price float(4,2)
);

create table order_items
( orderid int unsigned not null,
  isbn char(13) not null,
  quantity tinyint unsigned,

  primary key (orderid, isbn)

);
create table book_reviews
(
  isbn char(13) not null primary key,
  review text
);
```

Cada uma das tabelas é criada por uma instrução CREATE TABLE separada. Você pode ver que criamos cada uma das tabelas no esquema com as colunas que projetamos no último capítulo. Você verá que cada uma das colunas tem um tipo de dados listado depois de seu nome. Algumas colunas também têm outros especificadores.

O que as outras palavras-chave significam

NOT NULL significa que todas as linhas na tabela devem ter um valor nesse atributo. Se ele não for especificado, o campo pode estar em branco (NULL).

AUTO_INCREMENT é um recurso do MySQL especial que pode ser utilizado em colunas de números inteiros. Isso significa que se deixarmos aquele campo em branco ao inserir linhas na tabela, o MySQL automaticamente gerará um valor identificador único. O valor será um maior que o valor máximo já existente na coluna. Você só pode ter uma dessas em cada tabela. As colunas que especificam AUTO_INCREMENT devem ser indexadas.

PRIMARY KEY depois de um nome de coluna especifica que essa coluna é a chave primária para a tabela. As entradas nessa coluna precisam ser únicas. O MySQL indexará automaticamente essa coluna. Note que onde a utilizamos anteriormente com customerid na tabela customers, a utilizamos com AUTO_INCREMENT. O índice automático na chave primária cuida do índice exigido por AUTO_INCREMENT.

Especificar PRIMARY KEY depois de um nome de coluna só pode ser utilizado para uma chave primária exclusiva de coluna. A cláusula PRIMARY KEY no final da instrução order_items é uma forma alternativa. Nós a utilizamos aqui porque a chave primária para essa tabela consiste nas duas colunas conjuntamente.

UNSIGNED depois de um tipo inteiro significa que ele só pode ter um zero ou valor positivo.

Entendendo os tipos de coluna

Vamos tomar a primeira tabela como um exemplo:

```
create table customers
( customerid int unsigned not null auto_increment primary key,
  name char(30) not null,
  address char(40) not null,
  city char(20) not null
);
```

Ao criar qualquer tabela, você precisa tomar decisões sobre os tipos de coluna.

Com a tabela customers, temos quatro colunas da maneira como especificada no esquema. Primeiro, customerid é a chave primária, que especificamos diretamente. Decidimos que isso seria um inteiro (tipo de dados int) e que esses IDs deveriam ser unsigned. Além disso, tiramos proveito dos recursos de auto_increment para que o MySQL possa gerenciá-los por nós – menos um item com que se preocupar.

Todas as outras colunas armazenarão dados de tipo de string. Escolhemos o tipo char para essas. Isso especifica campos de largura fixa. A largura é especificada nos colchetes, então, por exemplo, name pode ter até 30 caracteres.

Esse tipo de dados sempre alocará 30 caracteres de armazenamento para o nome, mesmo que nem todos eles sejam utilizados. O MySQL preencherá os dados com espaços para deixá-los do tamanho certo. A alternativa é varchar, que utiliza só a quantidade de armazenamento exigido (mais um byte). É uma pequena troca – varchars utilizarão menos espaço mas chars são mais rápidos.

Observe que declaramos todas as colunas como NOT NULL. Essa é uma otimização menor que você pode fazer onde for possível para agilizar as coisas. Veremos otimização com mais detalhes no Capítulo 12.

Algumas das outras instruções CREATE têm variações na sintaxe. Vejamos a tabela orders:

```
create table orders
( orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null,
  amount float(6,2),
  date date not null
);
```

A coluna amount é especificada como um número flutuante de ponto de tipo float. Com a maioria dos tipos de dados de ponto flutuante, você pode especificar a largura de exibição e o número de casas decimais. Nesse caso, a quantidade de pedido estará em dólares, então permitimos um total de pedido razoavelmente grande (largura 6) e duas casas decimais para os centavos.

A coluna date tem o tipo de dados date.

Nessa tabela particular, especificamos que todas as colunas barrem a quantidade como NOT NULL. Por quê? Quando um pedido é inserido no banco de dados, precisaremos criá-lo em orders, adicionar os itens a order_items e então elaborar a quantidade. Talvez não saibamos a quantidade quando o pedido é criado, então permitimos que ela seja NULL.

A tabela books tem algumas características semelhantes:

```
create table books
(  isbn char(13) not null primary key,
   author char(30),
   title char(60),
   price float(4,2)
);
```

Nesse caso, não precisamos gerar a chave primária porque os ISBNs são gerados em outra parte. Temos à esquerda os outros campos NULL porque uma livraria talvez saiba o ISBN de um livro antes de ela saber title, author ou price. A tabela order_items demonstra como criar chaves primárias com diversas colunas:

```
create table order_items
( orderid int unsigned not null,
  isbn char(13) not null,
  quantity tinyint unsigned,

  primary key (orderid, isbn)
);
```

A tabela especifica a quantidade de um livro particular como um TINYINT UNSIGNED, que armazena um inteiro entre 0 e 255.

Como mencionamos antes, a chave primária de múltiplas colunas precisa ser especificada com uma cláusula especial de chave primária. Essa é utilizada aqui.

Por fim, considere a tabela book_reviews:

```
create table book_reviews
(
  isbn char(13) not null primary key,
  review text
);
```

Essa tabela utiliza um novo tipo de dados, text, que ainda não discutimos. Esse é utilizado para texto mais longo, como um artigo. Há algumas variantes nesse tipo de dados, que discutiremos mais adiante neste capítulo.

Para entender como criar tabelas em mais detalhe, vamos discutir nomes de coluna e identificadores em geral e depois os tipos de dados que podemos escolher para colunas. Então, primeiramente, vejamos o banco de dados que criamos.

Observando o banco de dados com SHOW e DESCRIBE

Efetue logon no monitor do MySQL e utilize o banco de dados de livros. Você pode visualizar as tabelas no banco de dados digitando:

```
mysql> show tables;
```

O MySQL exibirá uma lista de todas as tabelas no banco de dados:

```
+-----------------+
| Tables in books |
+-----------------+
| book_reviews    |
| books           |
| customers       |
| order_items     |
| orders          |
+-----------------+
5 rows in set (0.06 sec)
```

Você também pode utilizar show para ver uma lista de bancos de dados digitando:

```
mysql> show databases;
```

Você pode ver informações adicionais sobre uma tabela particular, por exemplo, books, utilizando DESCRIBE:

```
mysql> describe books;
```

O MySQL exibirá as informações que você forneceu ao criar o banco de dados:

```
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| isbn   | char(13)   |      | PRI |         |       |
| author | char(30)   | YES  |     | NULL    |       |
| title  | char(60)   | YES  |     | NULL    |       |
| price  | float(4,2) | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Esses comandos são úteis para lembrar-se de um tipo de coluna ou para navegar um banco de dados que você não criou.

Criando índices

Já mencionamos rapidamente índices, porque designar chaves primárias cria índices naquelas colunas.

Um problema comum encontrado por usuários novos MySQL é que eles reclamam do desempenho ruim desse banco de dados (que eles antes tinham ouvido falar que era bastante rápido). Esse problema de desempenho ocorre porque eles não criaram índices no banco de dados. (É possível criar tabelas sem chaves primárias ou índices.)

Para começar, os índices que foram criados automaticamente servirão. Se você achar que está executando muitas consultas em uma coluna que não é uma chave, pode querer adicionar um índice naquela coluna a fim de melhorar o desempenho. Para isso, utilize a instrução CREATE INDEX. O formato geral dessa instrução é:

```
CREATE [UNIQUE|FULLTEXT] INDEX nome_do_indice
ON nome_da_tabela (nome_da_coluna_do_indice [(tamanho)] [ASC|DESC], ...])
```

(Os índices FULLTEXT são para indexar campos de texto; discutimos seu uso no Capítulo 13.)

O campo opcional *tamanho* permite especificar que apenas os primeiros caracteres *tamanho* do campo serão indexados. Você também pode especificar se o índice deve ser ascendente (ASC) ou descendente (DESC); o default é ascendente.

Uma observação sobre tipos de tabelas

Você deve estar ciente de que o MySQL oferece mais de um tipo de tabela ou mecanismo de armazenamento, incluindo alguns tipos de transação segura. Discutimos os tipos de tabela no Capítulo 13. Atualmente, todas as tabelas no banco de dados utilizam o mecanismo padrão de armazenamento: MyISAM.

Entendendo identificadores do MySQL

Há cinco tipos de identificadores em MySQL – bancos de dados, tabelas, colunas e índices, que conhecemos, e aliases, que abordaremos no próximo capítulo.

Bancos de dados no MySQL mapeiam para diretórios na estrutura subjacente dos arquivos; e tabelas mapeiam para arquivos. Isso tem um efeito direto sobre os nomes que eles podem receber. Também afeta a distinção entre maiúsculas e minúsculas desses nomes – se os nomes de diretório e arquivo distinguem letras maiúsculas de minúsculas no seu sistema operacional, os nomes dos bancos de dados e tabelas distinguirão entre maiúsculas e minúsculas (por exemplo, em UNIX); caso contrário, eles não distinguirão (por exemplo, sob Windows). Os nomes de coluna e de alias não fazem distinção entre letras maiúsculas e minúsculas, mas você não pode utilizar versões com caixas diferentes na mesma instrução de SQL.

Como uma nota paralela, a localização do diretório e dos arquivos contendo os dados estará onde quer que ela tenha sido definida na configuração. Você pode verificar a localização no sistema utilizando o recurso mysqladmin da seguinte maneira:

```
mysqladmin variables
```

Depois procure a variável datadir.

Um resumo de possíveis identificadores é mostrado na Tabela 9.4. A única exceção adicional é que você não pode utilizar ASCII (0), ASCII(255) ou o caractere de aspas em identificadores (e para ser honesto, não estou seguro da razão pela qual você deveria).

Tabela 9.4 **Identificadores do MySQL**

Tipo	Comprimento máximo	Distinção entre maiúsculas e minúsculas?	Caracteres permitidos
Banco de dados	64	a mesma que o O/S	Qualquer caractere permitido em um nome de diretório no seu O/S, exceto /, \ e .
Tabela	64	a mesma que o O/S	Qualquer caractere permitido em um nome de arquivo no seu O/S exceto / e .
Coluna	64	não	Qualquer caractere.
Índice	64	não	Qualquer caractere.
Alias	255	não	Qualquer caractere.

Essas regras são extremamente abertas.

Como no MySQL 3.23.6, você mesmo pode ter palavras reservadas e caracteres especiais de todos os tipos nos identificadores, a única limitação é que se você utiliza algo estranho desse modo, tem de colocá-lo entre acentos graves (`) (esse acento grave encontra-se na mesma tecla que o sinal til no canto superior esquerdo do teclado padrão). Por exemplo:

```
create database `create database`;
```

As regras em versões do MySQL (antes da versão 3.23.6) são mais restritivas e não permitem que isso seja feito.

Naturalmente, você deve aplicar o senso comum a toda essa liberdade. Simplesmente porque você *pode* chamar um banco de dados `create database`, não significa que *deva fazê-lo*. O mesmo princípio se aplica a qualquer outro tipo de programação – utilize identificadores significativos.

Escolhendo tipos de dados de coluna

Os três tipos básicos de coluna no MySQL são: numérica, data e hora, e string. Dentro de cada uma dessas categorias está uma grande variedade de tipos. Resumiremos esses tipos aqui e entraremos em mais detalhe sobre os pontos fortes e fracos de cada um no Capítulo 12.

Cada um dos três tipos tem vários tamanhos de armazenamento. Quando escolher um tipo de coluna, geralmente o princípio é escolher o menor tipo em que os dados se ajustarão.

Para muitos tipos de dados, quando você estiver criando uma coluna desse tipo, pode especificar o comprimento máximo de exibição. Isso é mostrado nas tabelas de tipos de dados a seguir como `M`. Se ele for opcional para esse tipo, é mostrado entre colchetes. O valor máximo que você pode especificar para `M` é 255.

Valores opcionais por todas essas descrições são mostrados entre colchetes.

Tipos numéricos

Os tipos numéricos são inteiros ou números de ponto flutuante. Para os números de ponto flutuante, você pode especificar o número de dígitos depois da casa decimal. Isso é mostrado neste livro como `D`. O valor máximo que você pode especificar para `D` é 30 ou `M-2` (isto é, o comprimento máximo de exibição menos dois – um caractere para um ponto de fração decimal e um para a parte integral do número), qualquer que seja o mais baixo.

Para tipos de inteiro você também pode especificar se deseja que eles sejam `UNSIGNED`, como mostrado na Listagem 9.1.

Para todos os tipos numéricos, você também pode especificar o atributo `ZEROFILL`. Quando valores de uma coluna `ZEROFILL` forem exibidos, eles serão preenchidos com zeros iniciais. Se você especificar uma coluna como `ZEROFILL`, ela também será automaticamente `UNSIGNED`.

Os tipos integrais são mostrados na Tabela 9.5. Observe que os intervalos mostrados nessa tabela mostram o intervalo com sinal em uma linha e o intervalo sem sinal na seguinte.

Tabela 9.5 **Tipos de dados integrais**

Tipo	Faixa	Armazenamento (bytes)	Descrição
TINYINT[(*M*)]	–127..128 ou 0..255	1	Inteiros muito pequenos
BIT			Sinônimo de TINYINT
BOOL			Sinônimo de TINYINT
SMALLINT[(M)]	–32769..32767 ou	2	Inteiros pequenos
MEDIUMINT[(M)]	–8388609.. 8388607 ou 0..16777215	3	Inteiros de tamanho médio
INT[(M)]	–231..231-1 ou 0..232 –1	4	Inteiros regulares
INTEGER[(M)]			Sinônimo de INT
BIGINT[(M)]	$-2^{63}..2^{63}-1$ ou $0..2^{64}-1$	8	Inteiros grandes

O tipos de ponto flutuante são mostrados na Tabela 9.6.

Tabela 9.6 **Tipos de dados de ponto flutuante**

Tipo	Faixa	Armazenamento (bytes)	Descrição
FLOAT(*precisão*)	Depende da precisão	Varia	Pode ser utilizado para especificar números de ponto flutuante de precisão dupla ou simples.
FLOAT[(M,D)]	1.175494351E-38 ±3.402823466E+38	4	Número de ponto flutuante de precisão simples. Esses são equivalentes de FLOAT(4), mas com uma largura de exibição especificada e número de casas decimais.
DOUBLE[(M,D)]	±1.79769313486231 57E+308 ±2.22507385850720 14E -308	8	Número de ponto flutuante de dupla precisão. Esses são equivalentes de FLOAT(8), mas com uma largura de exibição e um número de casas decimais.
DOUBLE PRECISION[(M,D)]	Como anteriormente		Sinônimo de DOUBLE[(M, D)].
REAL[(M,D)]	Como anteriormente		Sinônimo de DOUBLE[(M, D)].
DECIMAL[(M[,D])]	Varia	M+2	Número de ponto flutuante armazenado como char. O intervalo depende de M, a largura de exibição.
NUMERIC[(M,D)]	Como anteriormente		Sinônimo de DECIMAL.
DEC[(M,D)]	Como anteriormente		Sinônimo de DECIMAL.
FIXED [(M,D)]	Como anteriormente		Sinônimo de DECIMAL.

Tipos de data e hora

O MySQL suporta vários tipos de data e hora. Essas classes são mostradas na Tabela 9.7. Com todos esses tipos, você pode inserir dados em uma string ou no formato numérico. Vale notar que uma coluna TIMESTAMP em uma linha particular será configurada como a data e a hora da operação mais recente nessa linha se você não configurá-la manualmente. Isso é útil para registro de transação.

Tabela 9.7 **Tipos de dados de data e hora**

Tipo	Faixa	Descrição
DATE	1000-01-01 9999-12-31	Uma data. Será exibido como YYYY-MM-DD.
TIME	-838:59:59 838:59:59	Um horário. Será exibido como HH:MM:SS. Observe que o intervalo tem uma largura muito maior que aquela que você provavelmente utilizaria algum dia.
DATETIME	1000-01-01 00:00:00 9999-12-31 23:59:59	Uma data e hora. Será exibido como YYYY-MM-DD HH:MM:SS.

Tabela 9.7 **Continuação**

Tipo	Faixa	Descrição
TIMESTAMP[(*M*)]	1970-01-01 00:00:00	Um registro de data/hora, útil para relatório de transação. O formato de exibição depende do valor de *M* (Ver Tabela 9.8, a seguir).
	Algum momento em 2037	A parte superior do intervalo depende do limite no UNIX.
YEAR[(*2*|*4*)]	70–69 (1970–2069) 1901–2155	Um ano. Você pode especificar formato de 2 ou 4 dígitos. Cada um desses tem um intervalo diferente, como mostrado.

A Tabela 9.8 mostra os diferentes tipos de exibição possíveis parTIMESTAMP (registro de data/hora).

Tabela 9.8 **Tipos de exibição de TIMESTAMP**

Tipo especificado	Exibição
TIMESTAMP	YYYYMMDDHHMMSS
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Tipos de string

Tipos de string dividem-se em três grupos. Primeiro, são strings velhas simples, isto é, pequenas partes de texto. Essas partes são os tipos CHAR (caractere de comprimento fixo) e VARCHAR (caractere de comprimento variável). Você pode especificar a largura de cada um. As colunas de tipo CHAR serão preenchidas com espaços até a largura máxima independente do tamanho dos dados, enquanto as colunas VARCHAR variam em largura com os dados. (Note que o MySQL distribuirá em faixas os espaços finais a partir de CHARs quando eles forem *recuperados* e a partir de VARCHARs quando eles forem *armazenados.)* Há uma relação de troca ou compensação entre espaço *versus* velocidade com esses dois tipos, o que discutiremos em mais detalhes no Capítulo 12.

O segundo, são os tipos TEXT e BLOB. Esses tipos ocorrem em vários tamanhos. Esses são para texto mais longo ou dados binários, respectivamente. BLOBs são *binary large objects* (objetos binários grandes). Esses podem armazenar o que você quiser, por exemplo, dados de imagem ou dados de som.

Na prática, as colunas BLOB e TEXT são as mesmas exceto que TEXT diferencia letras maiúsculas de minúsculas e BLOB não diferencia. Como esses tipos de coluna podem armazenar grandes quantidades de dados, eles requerem algumas considerações especiais. Discutiremos isso no Capítulo 12.

O terceiro grupo tem dois tipos especiais, SET e ENUM. O tipo SET é utilizado para especificar que valores nessa coluna devem vir de um conjunto particular de valores especificados. Os valores de coluna podem conter mais de um valor do conjunto. Você pode ter um máximo de 64 itens no conjunto especificado.

ENUM é uma enumeração. É muito semelhante a SET, exceto que colunas desse tipo podem ter somente um dos valores especificados ou NULL e um máximo de 65.535 itens na enumeração.

Resumimos os tipos de dados de string nas Tabelas 9.9, 9.10 e 9.11. A Tabela 9.9 mostra os tipos simples de string.

Tabela 9.9 **Tipos de string regular**

Tipo	Faixa	Descrição
[NATIONAL] CHAR(*M*) [BINARY | ASCII | UNICODE]	0 a 255 caracteres	String de comprimento fixo de comprimento *M*, onde *M* está entre 0 e 255. A palavra-chave NATIONAL especifica que o conjunto de caracteres padrão deve ser utilizado. De qualquer maneira, esse é o padrão em MySQL, mas ele é incluído como parte do padrão ANSI SQL. A palavra-chave BINARY especifica que os dados devem ser tratados como dados que não fazem distinção entre letras maiúsculas de minúsculas. (O padrão é fazer essa distinção.)
CHAR	1	Sinônimo de CHAR(1).
[NATIONAL] VARCHAR(*M*) [BINARY]	1 a 255 caracteres	Idêntico à descrição anterior, exceto que eles têm comprimento variável.

A Tabela 9.10 mostra os tipos TEXT e BLOB. O comprimento máximo de um campo TEXT em caracteres é o tamanho máximo em bytes de arquivos que poderiam ser armazenados nesse campo.

Tabela 9.10 **Tipos TEXT e BLOB**

Tipo	Comprimento máximo (caracteres)	Descrição
TINYBLOB	2^8-1 (isto é, 255)	Um campo BLOB pequeno.
TINYTEXT	2^8–1 (isto é, 255)	Um campo TEXT pequeno.
BLOB	2^{16}–1 (isto é, 65.535)	Um campo BLOB com tamanho normal.
TEXT	2^{16}–1 (isto é, 65.535)	Um campo TEXT com tamanho normal.
MEDIUMBLOB	2^{24}–1 (isto é, 16.777.215)	Um campo BLOB de tamanho médio.
MEDIUMTEXT	2^{24}–1 (isto é, 16.777.215)	Um campo TEXT de tamanho médio.
LONGBLOB	2^{32}–1 (isto é, 4.294.967.295)	Um campo BLOB longo.
LONGTEXT	2^{32}–1 (isto é, 4.294.967.295)	Um campo TEXT longo.

A Tabela 9.11 mostra os tipos ENUM e SET.

Tabela 9.11 **Tipos SET e ENUM**

Tipo	Valores máximos em Set	Descrição
ENUM('*valor1*', '*valor2*',...)	65.535	Colunas desse tipo só podem armazenar um dos valores listados ou NULL.
SET('*valor1*', '*valor2*',...)	64	Colunas desse tipo podem armazenar um conjunto de valores especificados ou NULL.

Leitura adicional

Para obter informações adicionais, você pode ler sobre como configurar um banco de dados no manual on-line MySQL em http://www.mysql.com/.

A seguir

Agora que você sabe criar usuários, bancos de dados e tabelas, pode se concentrar em interagir com o banco de dados. No próximo capítulo, veremos como colocar dados nas tabelas, como atualizar e excluir e como consultar o banco de dados.

10

Trabalhando com o banco de dados do MySQL

NESTE CAPÍTULO, DISCUTIREMOS Structured Query Language(SQL) e sua utilização em consulta de bancos de dados. Continuaremos desenvolvendo o banco de dados Book-O-Rama examinando como inserir, excluir e atualizar dados e como solicitar as perguntas de banco de dados.

Os tópicos que abordaremos incluem:

- O que é SQL?;
- Inserindo dados no banco de dados;
- Recuperando dados do banco de dados;
- Unindo tabelas;
- Utilizando subconsultas;
- Atualizando registros do banco de dados;
- Alterando tabelas depois da criação;
- Excluindo registros a partir do banco de dados;
- Excluindo tabelas.

Começaremos discutindo o que é a SQL e por que é útil entendê-la.

Se não configurou o banco de dados Book-O-Rama, você precisará fazer isso antes de poder executar as consultas de SQL neste capítulo. As instruções para fazer isso estão no Capítulo 9.

O que é SQL?

SQL significa *Structured Query Language*. Essa é a linguagem padrão para acessar *sistemas de gerenciamento de banco de dados relacional (relational database management systems – RDBMS)*. A SQL é utilizada para armazenar e recuperar dados para e a partir de um banco de dados. É utilizada em sistemas de banco de dados como MySQL, Oracle, PostgreSQL, Sybase e Microsoft SQL Server entre outros.

Há um padrão ANSI para SQL; e sistemas de banco de dados como MySQL geralmente se esforçam para implementar esse padrão. Há algumas diferenças sutis entre a SQL padrão e a SQL do MySQL. Algumas dessas diferenças são planejadas para se tornarem padrão em versões futuras de MySQL e algumas são deliberadas. Indicaremos as mais importantes à medida que avançarmos. Uma lista completa das diferenças entre a SQL do MySQL e o ANSI SQL em qual-

quer versão pode ser encontrada no manual on-line do MySQL. Você pode localizar essa página neste URL e em muitos outros locais:

http://www.mysql.com/doc/en/Compatibility.html

Talvez você tenha ouvido a expressão *linguagens de definição de dados (data definition language – DDL)*, utilizada para definir bancos de dados, e a expressão *linguagens de manipulação de dados (data manipulation language – DML)*, utilizada para consultar bancos de dados. A SQL abrange essas duas bases. No Capítulo 9, examinamos a definição de dados (DDL) na SQL, então já a utilizamos um pouco. Você utiliza DDL ao iniciar a configuração de um banco de dados.

Você utilizará os aspectos de DML da SQL muito mais freqüentemente porque essas são as partes que utilizamos para armazenar e recuperar dados reais em um banco de dados.

Inserindo dados no banco de dados

Antes de poder utilizar bastante um banco de dados, você precisa armazenar alguns dados nele. A maneira mais comum de fazer isso é com a instrução `INSERT` de SQL.

Lembre-se de que os RDBMSs contêm tabelas, que por sua vez contêm linhas de dados organizados em colunas. Normalmente, cada linha em uma tabela descreve algum objeto ou relacionamento do mundo real e os valores de coluna para essa linha armazenam informações de linha sobre o objeto do mundo real. Podemos utilizar a instrução `INSERT` para colocar linhas de dados no banco de dados.

A forma normal de uma instrução `INSERT` é:

```
INSERT [INTO] tabela[(coluna1, coluna2, coluna3,...)] VALUES
(valor1, valor2, valor3,...);
```

Por exemplo, para inserir um registro na tabela `Customers` de Book-O-Rama, você poderia digitar:

```
insert into customers values
  (NULL, "Julie Smith", "25 Oak Street", "Airport West");
```

Você pode ver que substituímos *`tabela`* pelo nome da tabela real na qual queremos colocar os dados e *`valores`* pelos valores específicos. Todos os valores nesse exemplo estão incluídos nas aspas duplas. As strings sempre devem ser incluídas em pares de aspas simples ou duplas no MySQL. (Utilizaremos as duas neste livro.) Os números e datas não precisam de aspas.

Há alguns aspectos interessantes sobre a instrução `INSERT` a serem notados. Os valores que especificamos serão utilizados para preencher as colunas de tabela em ordem. Se você quer preencher somente algumas colunas ou especificá-las em uma ordem diferente, pode listar as colunas específicas na parte de colunas da instrução. Por exemplo:

```
insert into customers (name, city) values
("Melissa Jones", "Nar Nar Goon North");
```

Essa abordagem é útil se você tiver apenas dados parciais sobre um registro particular ou se alguns campos no registro forem opcionais. Você também pode alcançar o mesmo efeito com a seguinte sintaxe:

```
insert into customers
set name="Michael Archer",
    address="12 Adderley Avenue",
    city="Leeton";
```

Você também notará que especificamos um valor `NULL` para a coluna `customerid` ao adicionar Julie Smith, e ignoramos essa coluna ao adicionar os outros clientes. Talvez você se lembre de que, quando configuramos o banco de dados, criamos `customerid` como a chave primária para a tabela `Customers`; portanto, isso poderia parecer estranho. Mas especificamos o campo como `AUTO_INCREMENT`. Isso significa que, se inserirmos uma linha com um valor `NULL` ou não inserirmos ne-

nhum valor nesse campo, o MySQL gerará o próximo número na seqüência de auto-incremento e o inserirá automaticamente para nós. Isso é muito útil.

Você também pode inserir várias linhas em uma tabela de uma vez. Cada linha deve estar no próprio conjunto de parênteses e cada conjunto de parênteses deve ser separado por uma vírgula.

Apenas algumas outras variações são possíveis com `INSERT`. Depois da palavra `INSERT`, você pode adicionar `LOW_PRIORITY` ou `DELAYED`. A palavra-chave `LOW_PRIORITY` significa que o sistema pode esperar e inserir posteriormente, quando os dados não estão sendo lidos da tabela. A palavra-chave `DELAYED` significa que os dados inseridos serão armazenados em buffer. Se o servidor estiver ocupado, você pode continuar a executar as consultas em vez de esperar que a operação `INSERT` complete.

Logo depois, você pode especificar opcionalmente `IGNORE`. Isso significa que se você tentar inserir linhas que geram uma chave única duplicada, elas serão ignoradas. Outra alternativa é especificar `ON DUPLICATE KEY UPDATE` *expressao* no final da instrução `INSERT`. Isso pode ser usado para mudar o valor duplicado utilizando uma instrução normal `UPDATE` (vista posteriormente neste capítulo).

Juntamos alguns dados simples de exemplo para preencher o banco de dados. Essa é apenas uma série de instruções `INSERT` simples que utilizam essa abordagem de inserção de várias linhas. O script que faz isso pode ser localizado no CD que acompanha este livro no arquivo `\chapter10\book_insert.sql`. Também é mostrado na Listagem 10.1.

Listagem 10.1 **book_insert.sql – SQL para preencher as tabelas para Book-O-Rama**

```
use books;

insert into customers values
  (NULL, "Julie Smith", "25 Oak Street", "Airport West"),
  (NULL, "Alan Wong", "1/47 Haines Avenue", "Box Hill"),
  (NULL, "Michelle Arthur", "357 North Road", "Yarraville");

insert into orders values
  (NULL, 3, 610.98, "2000-04-02"),
  (NULL, 1, 410.99, "2000-04-15"),
  (NULL, 2, 74.98, "2000-04-19"),
  (NULL, 3, 24.99, "2000-05-01");

insert into books values
  ("0-672-31697-8", "Michael Morgan", "Java 2 for Professional Developers", 34.99),
  ("0-672-31745-1", "Thomas Down", "Installing Debian GNU/Linux", 24.99),
  ("0-672-31509-2", "Pruitt, et al.", "Teach Yourself GIMP in 24 Hours", 24.99),
  ("0-672-31769-9", "Thomas Schenk", "Caldera OpenLinux System Administration
    Unleashed", 410.99);

insert into order_items values
  (1, "0-672-31697-8", 2),
  (2, "0-672-31769-9", 1),
  (3, "0-672-31769-9", 1),
  (3, "0-672-31509-2", 1),
  (4, "0-672-31745-1", 3);

insert into book_reviews values
  ("0-672-31697-8", "Morgan's book is clearly written and goes well beyond
                    most of the basic Java books out there.");
```

Você pode executar esse script redirecionando-o por meio do MySQL da seguinte maneira:

```
>mysql -h host -u bookorama -p < book_insert.sql
```

Recuperando dados do banco de dados

O cavalo de força da SQL é a instrução SELECT. Ela é utilizada para recuperar dados de um banco de dados selecionando linhas que correspondem ao critério especificado de uma tabela. Há uma grande quantidade de opções e maneiras diferentes de utilizar a instrução SELECT.

A forma básica de SELECT é:

```
SELECT [opções] itens
[INTO detalhes_do_arquivo]
FROM tabelas
[ WHERE condições ]
[ GROUP BY tipo_de_grupo ]
[ HAVING definição_de_where ]
[ ORDER BY tipo_de_pedido ]
[LIMIT critérios_de_limite ]
[PROCEDURE nome_proc(argumentos)]
[opções_de_bloqueio]
;
```

Nas próximas seções, descreveremos cada cláusula da instrução. Mas, antes de tudo, vamos examinar uma consulta sem cláusulas opcionais, uma que seleciona alguns itens a partir de uma tabela particular. Em geral, esses itens são colunas da tabela. (Esses itens também podem ser os resultados de qualquer expressão do MySQL. Discutiremos alguns dos itens mais úteis mais tarde nesta seção.) Esta consulta lista o conteúdo das colunas name e city da tabela Customers:

```
select name, city
from customers;
```

Essa consulta tem a seguinte saída, assumindo que você inseriu os dados de exemplo a partir da Listagem 10.1 e as outras duas instruções INSERT de exemplo:

```
+-----------------+--------------------+
| name            | city               |
+-----------------+--------------------+
| Julie Smith     | Airport West       |
| Alan Wong       | Box Hill           |
| Michelle Arthur | Yarraville         |
| Melissa Jones   | Nar Nar Goon North |
| Michael Archer  | Leeton             |
+-----------------+--------------------+
```

Como você pode ver, obtivemos uma tabela que contém os itens que selecionamos – name e city – a partir da tabela que especificamos, Customers. Esses dados são mostrados para todas as linhas na tabela Customer.

Você pode especificar quantas colunas quiser a partir de uma tabela listando-as depois da palavra-chave select. Você também pode especificar alguns outros itens. Um item útil é o operador curinga, *, que corresponde a todas as colunas na(s) tabela(s) especificada(s). Por exemplo, para recuperar todas as colunas e todas as linhas da tabela order_items, utilizaríamos

```
select *
from order_items;
```

que fornecerá a seguinte saída:

```
+---------+---------------+----------+
| orderid | isbn          | quantity |
+---------+---------------+----------+
|       1 | 0-672-31697-8 |        2 |
|       2 | 0-672-31769-9 |        1 |
|       3 | 0-672-31769-9 |        1 |
|       3 | 0-672-31509-2 |        1 |
|       4 | 0-672-31745-1 |        3 |
+---------+---------------+----------+
```

Recuperando dados com critérios específicos

A fim de acessar um subconjunto de linhas em uma tabela, precisamos especificar alguns critérios de seleção. Você pode fazer isso com uma cláusula WHERE. Por exemplo,

```
select *
from orders
where customerid = 3;
```

selecionará todas as colunas da tabela de pedidos, mas somente as linhas com um customerid de 3. Eis a saída:

```
+---------+------------+--------+------------+
| orderid | customerid | amount | date       |
+---------+------------+--------+------------+
|       1 |          5 |  69.98 | 2000-04-02 |
|       4 |          5 |  24.99 | 2000-05-01 |
+---------+------------+--------+------------+
```

A cláusula WHERE especifica o critério utilizado para selecionar linhas particulares. Nesse caso, selecionamos linhas com um customerid de 5. O sinal de igual único é utilizado para testar a igualdade – note que isso é diferente do PHP e é fácil de se confundir quando você estiver usando os dois juntos.

Além de igualdade, o MySQL suporta um conjunto completo de operadores de comparação e expressões regulares. As que você utilizará mais comumente nas cláusulas WHERE são listadas na Tabela 10.1. Observe que essa não é uma lista completa – se você precisar de algo não listado aqui, verifique o manual do MySQL.

Tabela 10.1 **Operadores de comparação úteis para cláusulas WHERE**

Operador	Nome (Se aplicável)	Exemplo	Descrição
=	igualdade	customerid = 3	Testa se os dois valores são iguais.
>	maior que	amount > 60.00	Testa se um valor é maior que outro.
<	menor que	amount < 60.00	Testa se um valor é menor que outro.
>=	maior ou igual	amount >= 60.00	Testa se um valor é maior que ou igual a outro.
<=	menor ou igual	amount <= 60.00	Testa se um valor é menor que ou igual a outro.
!= or < >	não igual	quantity != 0	Testa se dois valores não são iguais.
IS NOT NULL	n/d	address is not null	Testa se o campo realmente contém um valor.
IS NULL	n/d	address is null	Testa se o campo não contém um valor.
BETWEEN	n/d	amount between0 and 60.00	Testa se um valor é maior que ou igual a um valor mínimo e menor que ou igual a um valor máximo.
IN	n/d	city in ("Carlton","Moe")	Testa se um valor está em um conjunto particular.
NOT IN	n/d	city not in("Carlton","Moe")	Testa se um valor não está em um conjunto.
LIKE	correspondência de padrão	name like ("Fred %")	Verifica se um valor corresponde a um padrão utilizando correspondência de padrão de SQL simples.

Tabela 10.1 Continuação

Operador	Nome (Se aplicável)	Exemplo	Descrição
NOT LIKE	correspondência de padrão	name not like ("Fred %")	Verifica se um valor não corresponde ao padrão.
REGEXP	expressão regular	name regexp	Verifica se um valor corresponde a uma expressão regular.

As últimas três linhas na tabela referem-se a LIKE e REGEXP. Esses dois operadores são formas de correspondência padrão.

LIKE utiliza a correspondência de padrão da SQL simples. Os padrões podem consistir em texto regular mais o caractere % (porcentagem) para indicar uma correspondência de curinga com qualquer número de caracteres, e o caractere _ (sublinhado) para correspondência de curinga com um único caractere.

A palavra-chave REGEXP é utilizada para correspondência de expressão regular. O MySQL utiliza expressões POSIX regulares. Em vez de REGEXP, você também pode utilizar RLIKE, que é um sinônimo. Expressões POSIX regulares também são utilizadas em PHP. Você pode ler mais sobre elas no Capítulo 4.

Você pode testar diversos critérios dessa maneira e uni-los com AND e OR. Por exemplo,

```
select *
from orders
where customerid = 3 or customerid=4;
```

Recuperando dados de diversas tabelas

Freqüentemente, para responder a uma pergunta a partir do banco de dados, você precisará utilizar dados de mais de uma tabela. Por exemplo, se quisesse saber quais clientes fizeram pedidos este mês, você precisaria ver as tabelas Customers e Orders. Se você também quisesse saber o que, especificamente, eles encomendaram, também precisaria ver a tabela Order_Items.

Esses itens estão em tabelas separadas porque se relacionam com objetos do mundo real separados. Esse é um dos princípios do bom projeto de banco de dados sobre os quais conversamos no Capítulo 8.

Para agrupar essas informações na SQL, você deve realizar uma operação chamada *join* ou *junção*. Isso significa simplesmente unir duas ou mais tabelas para seguir os relacionamentos entre os dados. Por exemplo, se quisermos ver os pedidos que a cliente Julie Smith fez, precisaremos ver a tabela Customers para localizar o CustomerID de Julie e então a tabela Orders para localizar pedidos com esse CustomerID.

Embora as junções sejam conceitualmente simples, elas têm uma das partes mais sutis e complexas da SQL. Vários tipos diferentes de junção são implementados no MySQL e cada um é utilizado para um propósito diferente.

Junções simples de duas tabelas

Vamos começar examinando alguma SQL para a consulta sobre Julie Smith sobre a qual acabamos de conversar:

```
select orders.orderid, orders.amount, orders.date
from customers, orders
where customers.name = 'Julie Smith'
and customers.customerid = orders.customerid;
```

A saída dessa consulta é:

```
+---------+--------+------------+
| orderid | amount | date       |
+---------+--------+------------+
|       2 | 410.99 | 2000-04-15 |
+---------+--------+------------+
```

Há algumas coisas a serem notadas aqui.

Antes de tudo, como são necessárias informações das duas tabelas para responder a essa consulta, listamos as duas tabelas.

Além disso, especificamos um tipo de junção, possivelmente sem conhecê-la. A vírgula entre os nomes das tabelas é equivalente a digitar `INNER JOIN` ou `CROSS JOIN`. Esse é um tipo de junção que às vezes também é referida como *full join* (junção completa) ou *produto cartesiano* das tabelas. Isso significa: "aceite as tabelas listadas e faça uma tabela grande. A tabela grande deve ter uma linha para cada possível combinação de linhas de cada uma das tabelas listadas, quer isso faça sentido ou não". Em outras palavras, obtemos uma tabela, que tem cada linha da tabela `Customers` correspondida a cada linha da tabela `Orders`, independente de um cliente particular ter feito um pedido particular.

Isso não faz muito sentido na maioria dos casos. Freqüentemente, o que queremos é ver as linhas que realmente correspondem, isto é, os pedidos feitos por um cliente particular que correspondam a esse cliente.

Alcançamos isso colocando uma *condição join* na cláusula `WHERE`. Esse é um tipo de instrução condicional especial que explica quais atributos mostram o relacionamento entre as duas tabelas. Nesse caso, nossa condição join foi:

```
customers.customerid = orders.customerid
```

que diz ao MySQL para somente colocar linhas na tabela de resultado se o `CustomerId` da tabela `Customers` corresponder ao `CustomerID` da tabela `Orders`.

Adicionando essa condição join à consulta, realmente convertemos a junção para um tipo diferente, chamada de uma *equi-join*.

Você notará também que a notação de ponto que utilizamos para esclarecer a partir de qual tabela vem uma coluna particular, isto é, `customers.customerid` refere-se à coluna `customerid` da tabela `Customers` e `orders.customerid` refere-se à coluna `customerid` da tabela Orders.

Essa notação de ponto é requerida se o nome de uma coluna for ambíguo, isto é, se ocorrer em mais de uma tabela.

Como uma extensão, ela também pode ser utilizada para nomes de coluna para remover a ambigüidade de bancos de dados diferentes. Nesse exemplo, usamos uma notação `tabela.coluna`. Você pode especificar o banco de dados com uma notação `bancodedados.tabela.coluna`, por exemplo, para testar uma condição como:

```
books.orders.customerid = other_db.orders.customerid
```

Você pode, porém, utilizar a notação de ponto para todas as referências de coluna em uma consulta. Essa pode ser uma boa idéia, particularmente depois que suas consultas começarem a ficar mais complexas. O MySQL não exige, mas isso torna suas consultas muito mais legíveis por humanos e passíveis de manutenção. Você notará que seguimos essa convenção no restante da consulta anterior, por exemplo, com o uso da condição:

```
customers.name = 'Julie Smith'
```

A coluna `name` ocorre apenas na tabela `customers`; portanto, na verdade, não é necessário especificar de que tabela ela é. O MySQL não confundirá. No entanto, para nós, humanos, `name` sozinho é vago; portanto, o significado da consulta fica mais claro quando especificamos como `customer.name`.

Unindo mais de duas tabelas

Unir mais de duas tabelas não é mais difícil que uma junção de duas tabelas. Como uma regra geral, você precisa unir tabelas em pares com condições de junção. Pense nisso como seguir relacionamentos entre os dados de tabela para tabela sucessivamente.

Por exemplo, se quisermos saber quais clientes encomendaram livros sobre Java (de modo que possamos enviar a eles informações sobre um novo livro de Java), precisamos rastrear esses relacionamentos por meio de algumas tabelas.

Precisamos localizar clientes que fizeram pelo menos um pedido que incluiu um order_item que seja um livro sobre Java. Para ir da tabela Customers para a tabela Orders, podemos utilizar o customerid como fizemos anteriormente. Para ir da tabela Orders para a tabela Order_Items, podemos utilizar o orderid. Para ir da tabela Order_Items para o livro específico na tabela Books, podemos utilizar o ISBN. Depois de fazer todos esses links, você pode testar os livros com título Java e retornar os nomes de clientes que compraram qualquer um desses livros.

Vejamos uma consulta que faz tudo isso:

```
select customers.name
from customers, orders, order_items, books
where customers.customerid = orders.customerid
and orders.orderid = order_items.orderid
and order_items.isbn = books.isbn
and books.title like '%Java%';
```

Essa consulta retornará a seguinte saída:

```
+-----------------+
| name            |
+-----------------+
| Michelle Arthur |
+-----------------+
```

Observe que rastreamos os dados por meio de quatro tabelas diferentes e para fazer isso com uma equi-join, precisamos de três condições diferentes de junções. Geralmente é verdadeiro que você precisa de uma condição de junção para cada par de tabelas que deve unir e, portanto, um total de condições de junção menor que o número total de tabelas que deseja unir. Essa regra geral pode ser útil para depurar consultas que não funcionam bem. Verifique suas condições de junção e certifique-se de que seguiu completamente o caminho do que você sabe para o que deseja saber.

Localizando linhas que não correspondem

O outro tipo principal de junção que você utilizará no MySQL é a *left join*.

Nos exemplos anteriores, você notará que apenas as linhas em que existia uma correspondência entre as tabelas foram incluídas. Às vezes queremos especificamente as linhas em que não há nenhuma correspondência – por exemplo, clientes que nunca fizeram um pedido, ou livros que nunca foram encomendados.

A maneira mais fácil de responder a esse tipo de pergunta no MySQL é utilizar uma left join. Uma left join localizará linhas em uma condição join especificada entre duas tabelas. Se não houver nenhuma linha de correspondência na tabela certa, uma linha será adicionada ao resultado que contém valores NULL na coluna certa.

Vejamos um exemplo:

```
select customers.customerid, customers.name, orders.orderid
from customers left join orders
on customers.customerid = orders.customerid;
```

Essa consulta de SQL utiliza left join para unir Customers a Orders. Você notará que a left join utiliza uma sintaxe ligeiramente diferente para a condição de junção – nesse caso, a condição de junção entra em uma cláusula ON especial da instrução SQL.

O resultado dessa consulta é:

```
+------------+-----------------+---------+
| customerid | name            | orderid |
+------------+-----------------+---------+
|          1 | Melissa Jones   |    NULL |
|          2 | Michael Archer  |    NULL |
|          3 | Julie Smith     |       2 |
|          4 | Alan Wong       |       3 |
|          5 | Michelle Arthur |       1 |
|          5 | Michelle Arthur |       4 |
+------------+-----------------+---------+
```

Essa saída nos mostra que não há orderids correspondentes para os clientes Melissa Jones e Michael Archer porque os orderids para esses clientes são NULLs.

Se quisermos ver somente os clientes que não encomendaram nada, podemos fazer isso verificando esses NULLs no campo de chave primária da tabela certa (nesse caso orderid) já que isso não deve ser NULL em nenhuma linha real:

```
select customers.customerid, customers.name
from customers left join orders
using (customerid)
where orders.orderid is null;
```

O resultado é:

```
+------------+----------------+
| customerid | name           |
+------------+----------------+
|          4 | Melissa Jones  |
|          5 | Michael Archer |
+------------+----------------+
```

Você também notará que utilizamos uma sintaxe diferente para a condição de junção nesse exemplo. As left joins suportam tanto a sintaxe ON que utilizamos no primeiro exemplo como a sintaxe USING no segundo exemplo. Note que a sintaxe USING não especifica a tabela de que provém o atributo join – por essa razão, as colunas nas duas tabelas devem ter o mesmo nome se você quiser utilizar USING.

Você também pode responder a esse tipo de questão utilizando subconsultas. Veremos subconsultas posteriormente neste capítulo.

Utilizando outros nomes para tabelas: aliases

A capacidade de referenciar tabelas com outros nomes costuma ser útil e ás vezes é essencial. Outros nomes para tabelas são chamados *aliases*. Você pode criar aliases no início de uma consulta e então utilizá-los em qualquer lugar. Eles são freqüentemente úteis como abreviação. Considere a enorme consulta que vimos antes, regravada com aliases:

```
select c.name
from customers as c, orders as o, order_items as oi, books as b
where c.customerid = o.customerid
and o.orderid = oi.orderid
and oi.isbn = b.isbn
and b.title like '%Java%';
```

Quando declaramos as tabelas que vamos utilizar, adicionamos uma cláusula AS para declarar o alias para essa tabela. Podemos também utilizar aliases para colunas, mas retornaremos para isso quando examinarmos as funções agregadas em um instante.

Precisamos utilizar aliases de tabela quando queremos unir uma tabela com ela própria. Isso soa mais difícil e esotérico do que é na realidade. Fazer isso é útil, se, por exemplo, quisermos localizar linhas na mesma tabela que têm valores em comum. Se quisermos localizar clientes que vivem na mesma cidade – talvez para configurar um grupo de leitura – podemos fornecer dois aliases diferentes para a mesma tabela (Customers):

```
select c1.name, c2.name, c1.city
from customers as c1, customers as c2
where c1.city = c2.city
and c1.name != c2.name;
```

O que basicamente estamos fazendo é fingir que a tabela Customers é duas tabelas diferentes, c1 e c2 e realizar uma junção na coluna City. Você notará que também precisamos da segunda condição, c1.name != c2.name – isso serve para evitar que cada cliente represente uma correspondência com ele próprio.

Resumo de joins

Os diferentes tipos de junção que vimos estão resumidos na Tabela 10.2. Há alguns outros, mas estes resumidos são os principais que você vai utilizar.

Tabela 10.2 **Tipos de junção no MySQL**

Nome	Descrição
Produto cartesiano	Todas as combinações de todas as linhas em todas as tabelas na junção. Utilizado especificando uma vírgula entre nomes de tabela e não especificando uma cláusula WHERE.
Full join	Mesma que a precedente.
Cross join	O mesmo que anteriormente. Também pode ser utilizada especificando as palavras-chave CROSS JOIN entre os nomes das tabelas que estão sendo unidas.
Inner join	Semanticamente equivalente à vírgula. Também pode ser especificada utilizando as palavras-chave INNER JOIN. Sem a condição WHERE, equivale a uma full join. Em geral, você também especificará uma condição WHERE para tornar essa uma inner join verdadeira.
Equi-join	Utiliza uma expressão condicional com um = para corresponder linhas de tabelas diferentes na junção. Na SQL, essa é uma junção com uma cláusula WHERE.
Left join	Tenta corresponder linhas entre tabelas e preencher linhas não-correspondentes com NULLs. Use em SQL com as palavras-chave LEFT JOIN. Utilizada para localizar valores ausentes. Você pode utilizar equivalentemente RIGHT JOIN.

Recuperando dados em uma ordem particular

Se quiser exibir linhas recuperadas por uma consulta em uma ordem particular, você pode utilizar a cláusula ORDER BY da instrução SELECT. Esse recurso é útil para apresentar saída em um bom formato legível por humano.

A cláusula ORDER BY é utilizada para classificar as linhas em uma ou mais das colunas listadas na cláusula SELECT. Por exemplo:

```
select name, address
from customers
order by name;
```

Essa consulta retornará nomes de cliente e endereços na ordem alfabética por nome, assim:

```
+-----------------+--------------------+
| name            | address            |
+-----------------+--------------------+
| Alan Wong       | 1/47 Haines Avenue |
| Julie Smith     | 25 Oak Street      |
| Melissa Jones   |                    |
| Michael Archer  | 12 Adderley Avenue |
| Michelle Arthur | 357 North Road     |
+-----------------+--------------------+
```

Note que nesse caso, como os nomes estão no formato primeiro nome, sobrenome, eles são alfabeticamente classificados pelo primeiro nome. Se quisesse classificar por sobrenomes, você precisaria colocá-los em dois campos diferentes.

A ordenação padrão é ascendente (de a a z ou numericamente crescente). Você pode especificar isso se quiser utilizando a palavra-chave `ASC`:

```
select name, address
from customers
order by name asc;
```

Você também pode fazer na ordem oposta utilizando a palavra-chave `DESC` (descendente):

```
select name, address
from customers
order by name desc;
```

Você pode classificar mais de uma coluna e também pode utilizar aliases de coluna ou mesmo seus números de posição (por exemplo, 3 é a terceira coluna na tabela) em vez de nomes.

Agrupando e agregando dados

Freqüentemente queremos saber quantas linhas caem dentro de um conjunto particular ou o valor médio de alguma coluna – digamos, o valor médio em dólares por pedido. O MySQL tem um conjunto de funções agregadas que são úteis para responder a esse tipo de consulta.

Essas funções agregadas podem ser aplicadas a uma tabela como a um total ou grupos de dados dentro de uma tabela.

As mais comumente utilizadas são listadas na Tabela 10.3.

Tabela 10.3 **Funções agregadas no MySQL**

Nome	Descrição
`AVG(`*coluna*`)`	Média de valores na coluna especificada.
`COUNT(`*itens*`)`	Se você especificar uma coluna, essa função fornecerá o número de valores não-`NULL` nessa coluna. Se adicionar a palavra `DISTINCT` na frente do nome de coluna, você obterá uma contagem dos valores distintos somente nessa coluna. Se especificar `COUNT(*)`, você obterá uma contagem de linha independente dos valores `NULL`.
`MIN(coiluna)`	O valor mínimo na coluna especificada.
`MAX(`*coluna*`)`	O valor máximo na coluna especificada.

Tabela 10.3 **Continuação**

Nome	Descrição
STD(*coluna*)	Desvio padrão dos valores na coluna especificada.
STDDEV(*coluna*)	Mesmo que STD(*coluna*).
SUM(*coluna*)	Soma de valores na coluna especificada.

Vejamos alguns exemplos, começando com o mencionado anteriormente. Podemos calcular o total médio de um pedido assim:

```
select avg(amount)
from orders;
```

A saída será algo assim:

```
+-------------+
| avg(amount) |
+-------------+
|   54.985002 |
+-------------+
```

A fim de obter informações mais detalhadas, podemos utilizar a cláusula GROUP BY. Isso permite visualizar o total médio de pedidos por grupo – por exemplo, por número de cliente. Isso nos dirá qual de nossos clientes fez os maiores pedidos:

```
select customerid, avg(amount)
from orders
group by customerid;
```

Quando você utiliza uma cláusula GROUP BY com uma função agregada, ela realmente altera o comportamento da função. Em vez de fornecer uma média da quantidade de pedidos da tabela, essa consulta fornecerá a quantidade média de pedidos para cada cliente (ou, mais especificamente, para cada customerid):

```
+------------+-------------+
| customerid | avg(amount) |
+------------+-------------+
|          1 |  410.990002 |
|          2 |   74.980003 |
|          3 |   47.485002 |
+------------+-------------+
```

Algo a ser observado ao utilizar agrupamento e funções agregadas: no ANSI SQL, se você utilizar uma função agregada ou a cláusula GROUP BY, as únicas coisas que podem aparecer na cláusula SELECT são funções agregadas e as colunas chamadas na cláusula GROUP BY. Além disso, se você quiser utilizar uma coluna em uma cláusula GROUP BY, ela deve ser listada na cláusula SELECT.

O MySQL realmente fornece um pouco mais de espaço aqui. O MySQL suporta uma *sintaxe estendida*, que permite deixar itens fora da cláusula SELECT se você realmente não quiser esses itens.

Além de agrupar e agregar dados, podemos realmente testar o resultado de um agregado utilizando uma cláusula HAVING. Isso vem logo depois da cláusula GROUP BY e é como uma WHERE que se aplica somente a grupos e agregados.

Para estender nosso exemplo anterior, se quisermos saber quais clientes têm um total médio de pedido de mais de US$50, você pode utilizar a seguinte consulta:

```
select customerid, avg(amount)
from orders
group by customerid
having avg(amount) > 50;
```

Observe que a cláusula HAVING se aplica aos grupos. Essa consulta retornará a seguinte saída:

```
+------------+-------------+
| customerid | avg(amount) |
+------------+-------------+
|          2 |   74.980003 |
+------------+-------------+
```

Escolhendo quais linhas retornar

Uma cláusula da instrução SELECT que pode ser particularmente útil nas aplicações Web é a cláusula LIMIT. Essa é utilizada para especificar quais linhas da saída devem ser retornadas. Essa cláusula aceita dois parâmetros: o número de linha a partir do qual iniciar e o número de linhas para retornar.

Essa consulta ilustra o uso de LIMIT:

```
select name
from customers
limit 2, 3;
```

Essa consulta pode ser lida como: "selecione *name* de *customers* e então retorne 3 linhas, iniciando da linha 2 na saída". Note que os números de linha são indexados a partir de zero – isto é, a primeira linha na saída é a linha número zero.

Isso é muito útil para aplicações Web, como quando o cliente está navegando por produtos em um catálogo e queremos mostrar dez itens em cada página. Observe, entretanto, que LIMIT não faz parte de ANSI SQL. É uma extensão MySQL, então usá-la faz seu SQL incompatível com os RDBMSs.

Utilizando subconsultas

Uma subconsulta é uma consulta que está dentro de outra consulta. Esse recurso é novo no MySQL 4.1. Embora a maior parte da funcionalidade das subconsultas possa ser obtida com o uso cuidadoso de joins e tabelas temporárias, as subconsultas são mais fáceis de ler e escrever.

Subconsultas básicas

A aplicação mais comum das subconsultas é usar o resultado de uma consulta comparando-o com o de outra. Por exemplo, se você quiser descobrir a quantidade máxima de todos os pedidos, poderia utilizar a seguinte consulta:

```
select customerid, amount
from pedidos
where amount = (select max(amount) from pedidos);
```

Essa consulta produz os seguintes resultados:

```
+------------+--------+
| customerid | amount |
+------------+--------+
|          4 |  74.98 |
+------------+--------+
```

Nesse caso, um único valor é retornado da subconsulta (a quantidade máxima) e então usado para comparação na consulta mais externa. Esse é um bom exemplo do uso de subconsulta porque não pode ser reproduzido de forma elegante utilizando joins em ANSI SQL.

A mesma saída, entretanto, produzida por esta consulta join:

```
select customerid, amount
from pedidos
order by amount desc
limit 1;
```

Como se baseia em LIMIT, essa consulta não é compatível com a maioria dos RDBMSs, mas é executado de forma mais eficiente no MySQL do que a versão da subconsulta.

Uma das razões por que o MySQL não aceitou subconsultas por tanto tempo é que há muito pouco que não é possível fazer sem elas. Tecnicamente, você pode criar uma única consulta SQL ANSI legal que tenha o mesmo efeito, mas que se baseie em um hack com uma abordagem ineficiente, chamado MAX-CONCAT.

Você pode utilizar valores de subconsulta nessa forma com todos os operadores de comparação normais. Alguns operadores de comparação de subconsulta também estão disponíveis, e são detalhados na próxima seção.

Subconsultas e operadores

Existem cinco operadores de subconsulta especiais. Quatro são usados com subconsultas regulares e um (EXISTS) geralmente é usado apenas com subconsultas correlacionadas, e será visto na próxima seção. Os quatro operadores de subconsulta regular são mostrados na Tabela 10.4.

Tabela 10.4 **Operadores de subconsulta**

Nome	Sintaxe de Exemplo	Descrição
ANY	SELECT c1 FROM T1 WHERE c1 > ANY (SELECT c1 FROM t2);	Retorna True se a comparação for verdadeira para qualquer linha na subconsulta.
IN	SELECT c1 FROM t1 WHERE c1 IN (SELECT c1 from t2);	Equivalente a =ANY.
SOME	SELECT c1 FROM t1 WHERE c1 > SOME (SELECT c1 FROM t2)	Alias para ANY; às vezes lê melhor para o ouvido humano.
ALL	SELECT c1 FROM t1 WHERE c1 > ALL 9SELECT c1 from t2);	Retorna true se a comparação for verdadeira para todas as linhas na subconsulta.

Cada um desses operatores pode aparecer apenas após um operador de comparação, exceto o IN, que já tem seu operador de comparação embutido (=).

Subconsultas correlacionadas

Em subconsultas correlacionadas, as coisas se complicam um pouco mais. Nelas, você pode utilizar itens da consulta externa na consulta interna. Por exemplo:

```
select isbn, title
from books
where not exists
 (select * from order_items where order_items.isbn=books.isbn);
```

Essa consulta ilustra o uso de subconsultas correlacionadas e o uso do último operador especial de subconsulta, EXISTS. Ela recupera os livros que nunca foram pedidos. (A mesma informação que você encontrou fazendo uma left join anteriormente.) Observe que a consulta interna inclui a tabela

order_items apenas na lista FROM, mas se refere a books.isbn. Em outras palavras, a consulta interna se refere as dados na consulta externa. Esta é a definição de uma subconsulta correlacionada: você está procurando por linhas internas que correspondam (ou nesse caso não correspondam) às linhas externas.

O operador EXISTS retorna true se houver linhas correspondentes na subconsulta. Da mesma forma, NOT EXISTS retorna true se não houver linhas correspondentes na subconsulta.

Subconsultas de linha

Todas as subconsultas até agora retornaram um único valor, embora, em muitos casos, esse valor seja true ou false (como no exemplo anterior, utilizando EXISTS). As subconsultas de linha retornam uma linha inteira, que pode, então, ser comparada às linhas inteiras na consulta externa. Essa abordagem geralmente é usada para procurar linhas em uma tabela que também existam em outra. Não existe um bom exemplo dela no banco de dados do livro, mas um exemplo generalizado da sintaxe pode se parecer com:

```
select c1, c2, c3
from t1
where (c1, c2, c3) in (select c1, c2, c3 from t2);
```

Utilizando uma subconsulta como tabela temporária

Você pode utilizar uma subconsulta na cláusula FROM de um consulta externa. Essa abordagem permite consultar a saída da subconsulta, tratando-a como uma tabela temporária.

Em seu formato mais simples, ela se pareceria com:

```
select * from
(select customerid, name from customers where city='Box Hill')
as box_hill_customers;
```

Observe que colocamos a subconsulta na cláusula FROM aqui. Imediatamente após o parêntese final da subconsulta, é preciso fornecer um alias aos resultados da subconsulta. Então, você pode tratá-la como qualquer outra tabela na consulta externa.

Atualizando registros no banco de dados

Além de recuperar dados a partir do banco de dados, freqüentemente queremos alterá-lo. Por exemplo, talvez queiramos aumentar os preços de livros no banco de dados. Podemos fazer isso utilizando uma instrução UPDATE.

A forma normal de uma instrução UPDATE é:

```
UPDATE [LOW_PRIORITY] [IGNORE] nomedatabela
SET coluna1=expressão1,coluna2=expressão2,...
[WHERE condição]
[ORDER BY critérios_de_pedidos]
[LIMIT número]
```

A idéia básica é atualizar a tabela chamada *nomedatabela*, configurando cada uma das colunas chamadas como a expressão apropriada. Você pode limitar UPDATE para linhas particulares com uma cláusula WHERE e limitar o número total de linhas para afetar com uma cláusula LIMIT. ORDER BY geralmente é usado junto com a cláusula LIMIT, por exemplo, se quisermos atualizar apenas as 10 primeiras linhas, temos de colocá-las em algum tipo de ordem primeiro.

LOW PRIORITY e IGNORE, se especificadas, funcionam da mesma maneira que a instrução INSERT.

Vamos ver alguns exemplos. Se você quiser aumentar 10% do preço de todos os livros, podemos utilizar uma instrução UPDATE sem a cláusula WHERE:

```
update books
set price=price*1.1;
```

Se, por outro lado, quisermos alterar uma única linha – digamos, atualizar um endereço do cliente – podemos fazer isso assim:

```
update customers
set address = '250 Olsens Road'
where customerid = 4;
```

Alterando tabelas depois da criação

Além de atualizar linhas, talvez você queira alterar a estrutura das tabelas dentro do seu banco de dados. Para esse propósito, você pode utilizar a flexível instrução `ALTER TABLE`. A forma básica dessa instrução é:

```
ALTER TABLE nomedatabela alteração [, alteração ...]
```

Observe que no ANSI você pode fazer somente uma alteração por instrução `ALTER TABLE`, mas o MySQL permite fazer quantas quiser. Cada uma das cláusulas de alteração pode ser utilizada para alterar diferentes aspectos da tabela.

Se a cláusula `IGNORE` for especificada e você estiver tentando fazer uma alteração que gere chaves primárias duplicadas, a primeira entrará na tabela alterada e o restante será excluído. Se não for especificada (o default), a alteração falhará e será retornada.

Os diferentes tipos de alteração que você pode fazer com essa instrução são mostrados na Tabela 10.5.

Tabela 10.5 **Possíveis alterações com a instrução ALTER TABLE**

Sintaxe	Descrição
ADD [COLUMN] *descrição_da_coluna* [FIRST | AFTER *coluna*]	Adiciona uma nova coluna no local especificado (se não for especificado, então a coluna entra no final). Observe que *descrição_da_coluna* precisa de um nome e um tipo, exatamente como na instrução CREATE.
ADD [COLUMN] (descrição_da_coluna, *descrição_da_coluna,...*)	Adiciona uma ou mais colunas no final da tabela.
ADD INDEX [*índice*] (*coluna,...*)	Adiciona um índice à tabela na(s) coluna(s) especificada(s).
ADD [CONSTRAINT [*símbolo*]] PRIMARY KEY_(*coluna,...*)	Torna a(s) coluna(s) especificada(s) a chave primária da tabela. A notação CONSTRAINT é para tabelas que utilizam chaves estrangeiras. Ver Capítulo 13 para mais detalhes.
ADD UNIQUE [CONSTRAINT [*símbolo*]] [*índice*] (*coluna,...*)	Adiciona um único índice à tabela na(s) coluna(s) especificada(s). A notação _CONSTRAINT é para tabelas InnoDB que utilizam chaves estrangeiras. Ver o Capítulo 13 para mais detalhes.
ADD [CONSTRAINT [*símbolo*]] FOREIGN KEY [*índice*] (*índice_col,...*) [*definição_da_referência*]	Adiciona uma chave estrangeira a uma tabela InnoDB. Ver o Capítulo 13 para mais detalhes.
ALTER [COLUMN] *coluna* {SET DEFAULT valor | DROP DEFAULT}	Adiciona ou remove um valor default para uma coluna específica.
CHANGE [COLUMN] *coluna* *nova_descrição_da_coluna*	Muda a coluna chamada *coluna* de modo que tenha a descrição listada. Observe que essa sintaxe pode ser usada para alterar o nome de uma coluna porque *descrição_da_coluna* inclui um nome.
MODIFY [COLUMN] descrição_da_coluna	Semelhante a CHANGE. Pode ser usado para alterar tipos de colunas, não nomes.

Tabela 10.5 Continuação

Sintaxe	Descrição
DROP [COLUMN] column	Exclui a coluna identificada.
DROP PRIMARY KEY	Exclui o índice primário (mas não a coluna).
DROP INDEX índice	Exclui o índice identificado.
DROP FOREIGN KEY *chave*	Exclui a chave estrangeira (mas não a coluna).
DISABLE KEYS	Desativa a atualização do índice.
ENABLE KEYS	Ativa a atualização de índice.
RENAME [AS] *novo_nome_de_tabela*	Renomeia uma tabela.
ORDER BY *nome_col*	Recria a tabela com as linhas em uma ordem específica. (Observe que depois de começar a mudar a tabela, as linhas não estarão mais em ordem.)
CONVERT TO CHARACTER SET *cs* COLLATE *c*	Converte todas as colunas baseadas em texto para o conjunto de caractere especificado.
[DEFAULT] CHARACTER SET *cs* COLLATE *c*	Define o conjunto de caracteres default.
DISCARD TABLESPACE	Exclui o arquivo de espaço de tabela para uma tabela InnoDB. (Ver o Capítulo 13 para mais detalhes sobre InnoDB.)
IMPORT TABLESPACE	Recria o arquivo de espaço de tabela para uma tabela InnoDB. (Ver o Capítulo 13 para mais detalhes sobre InnoDB.)
opções_da_tabela	Permite redefinir as opções da tabela. Utiliza a mesma sintaxe de CREATE TABLE.

Vejamos algumas das utilizações mais comuns de ALTER TABLE.

Algo que ocorre com frequência é perceber que a coluna criada não é "grande o suficiente" para os dados que têm de ser armazenados. Por exemplo, na tabela customers, permitimos que os nomes tenham 30 caracteres de comprimento. Depois que começamos a obter alguns dados, talvez notemos que alguns nomes são muito longos e que estão sendo truncados. Podemos corrigir isso alterando o tipo de dados da coluna para que, em vez disso, ela tenha 70 caracteres:

```
alter table customers
modify name char(70) not null;
```

Outra ocorrência comum é a necessidade de adicionar uma coluna. Imagine que um imposto de vendas sobre livros é introduzido localmente e que Book-O-Rama precisa adicionar a quantidade de imposto ao pedido total, mas monitorá-lo separadamente. Podemos adicionar uma coluna de imposto à tabela de pedidos da seguinte maneira:

```
alter table orders
add tax float(6,2) after amount;
```

Eliminar uma coluna é outro caso que surge freqüentemente. Podemos excluir a coluna que acabamos de adicionar da seguinte maneira:

```
alter table orders
drop tax;
```

Excluindo registros do banco de dados

Excluir linhas do banco de dados é muito simples. Você pode fazer isso utilizando a instrução DELETE, que é geralmente assim:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tabela
[WHERE condição]
[ORDER BY cols_de_pedidos]
[LIMIT número]
```

Se você escrever:

```
DELETE FROM tabela;
```

apenas, todas as linhas em uma tabela serão excluídas, então tenha cuidado! Normalmente, você quer excluir linhas específicas e pode especificar as que deseja excluir com uma cláusula WHERE. Talvez você faça isso, se, por exemplo, um livro particular não estiver mais disponível ou se um cliente particular não tiver feito qualquer pedido durante muito tempo e você quiser fazer uma faxina:

```
delete from customers
where customerid=5;
```

A cláusula LIMIT pode usada para limitar o número máximo de linhas que são realmente excluídas. ORDER BY é geralmente usado junto com LIMIT.

LOW_PRIORITY e IGNORE funcionam da mesma forma que em outras situações. QUICK pode ser mais rápido em tabelas MyISAM.

Excluindo tabelas

Às vezes, você pode querer se livrar de uma tabela inteira, e, para isso, utilize a instrução DROP TABLE. Esse processo é muito simples, e é semelhante a:

```
DROP TABLE tabela;
```

Essa consulta exclui todas as linhas na tabela e a própria tabela; portanto, tenha cuidado ao usá-la.

Excluindo um banco de dados inteiro

Você pode avançar ainda mais e eliminar um banco de dados inteiro com a instrução DROP DATABASE, que é parecida com:

```
DROP DATABASE banco_de_dados;
```

Isso excluirá todas as linhas, todas as tabelas, todos os índices e o próprio banco de dados, então continuo dizendo que você deve ser um pouco cuidadoso ao utilizar essa instrução.

Leitura adicional

Neste capítulo, fornecemos uma visão geral da SQL rotineira que você utilizará ao interagir com um banco de dados MySQL. Nos próximos dois capítulos, veremos como juntar MySQL e PHP para acessar o banco de dados Web. Também exploraremos algumas técnicas avançadas do MySQL.

Se quiser saber mais sobre a SQL, você pode sempre voltar ao ANSI SQL padrão para uma breve e rápida leitura. Ele está disponível em:

http://www.ansi.org/

Para obter mais detalhes sobre as extensões do MySQL para o ANSI SQL, você pode ver o Web site do MySQL :

http://www.mysql.com

A seguir

No Capítulo 11, abordaremos como você pode disponibilizar o banco de dados Book-O-Rama na Web.

11

Acessando o banco de dados MySQL a partir da Web com o PHP

ANTERIORMENTE, EM NOSSO TRABALHO COM O PHP, utilizamos um arquivo simples para armazenar e recuperar dados. Quando vimos isso no Capítulo 2, mencionamos que sistemas de banco de dados relacional tornam uma grande quantidade dessas tarefas de armazenamento e recuperação mais fáceis, mais seguras e mais eficientes em uma aplicação Web. Agora, tendo trabalhado com o MySQL para criar um banco de dados, podemos começar conectando esse banco de dados a um front end baseado na Web.

Neste capítulo, explicaremos como acessar o banco de dados Book-O-Rama a partir da Web utilizando o PHP. Você aprenderá a ler e gravar no banco de dados e a filtrar dados de entrada potencialmente problemáticos.

Em resumo, veremos:

- Como funcionam as arquiteturas dos bancos de dados Web;
- Os passos básicos na consulta de um banco de dados Web;
- Como configurar uma conexão;
- Como obter informações sobre bancos de dados disponíveis;
- Como escolher um banco de dados para utilização;
- Como consultar o banco de dados;
- Como recuperar resultados da consulta;
- Como se desconectar do banco de dados;
- Como inserir novas informações no banco de dados;
- Outras funções PHP–MySQL úteis;
- Como utilizar uma interface de banco de dados genérica: PEAR DB;
- Outras interfaces de banco de dados do PHP.

Como as arquiteturas do banco de dados Web funcionam

No Capítulo 8, descrevemos como as arquiteturas de banco de dados Web funcionam. Só para lembrar, eis os passos novamente:

1. Um navegador Web do usuário emite uma solicitação de HTTP para uma página Web particular. Por exemplo, o usuário poderia ter solicitado uma pesquisa de todos os livros escritos por Michael Morgan na Book-O-Rama, utilizando um formulário HTML. A página de resultados de pesquisa denomina-se `results.php`.
2. O servidor Web recebe a solicitação para `results.php`, recupera o arquivo e passa para o mecanismo de PHP para processamento.
3. O mecanismo de PHP começa a analisar sintaticamente o script. Dentro do script está um comando para conectar-se ao banco de dados e executar uma consulta (realizar a pesquisa dos livros). O PHP abre uma conexão com o servidor do MySQL e envia a consulta apropriada.
4. O servidor do MySQL recebe a consulta de banco de dados, a processa e envia os resultados – uma lista de livros – de volta para o mecanismo de PHP.
5. O mecanismo de PHP termina de executar o script que normalmente envolverá a formatação dos resultados da consulta em HTML de maneira elegante. Então, ele retorna a HTML resultante para o servidor Web.
6. O servidor Web passa a HTML de volta para o navegador, onde o usuário pode ver a lista de livros que ele solicitou.

Agora, temos um banco de dados do MySQL existente, então podemos escrever o código de PHP para realizar os passos anteriores. Iniciaremos com o formulário de pesquisa. Esse é um formulário HTML simples. O código para o formulário é mostrado na Listagem 11.1.

Listagem 11.1 **search.html – Página Database Search da Book-O-Rama**

```
<html>
<head>
  <title>Book-O-Rama Catalog Search</title>
</head>

<body>
  <h1>Book-O-Rama Catalog Search</h1>

  <form action="results.php" method="post">
    Choose Search Type:<br />
    <select name="searchtype">
      <option value="author">Author</option>
      <option value="title">Title</option>
      <option value="isbn">ISBN</option>
    </select>
    <br />
    Enter Search Term:<br />
    <input name="searchterm" type="text">
    <br />
    <input type="submit" value="Search">
  </form>

</body>
</html>
```

Esse é um formulário HTML muito simples e direto. A saída dessa HTML é mostrada na Figura 11.1.

Figura 11.1 O formulário de pesquisa é bem geral, permitindo procurar um livro pelo seu título, autor ou ISBN.

O script que será chamado quando o botão de pesquisa for pressionado é `results.php`. Esse botão é listado integralmente na Listagem 11.2. No decorrer deste capítulo, discutiremos o que esse script faz e como funciona.

Listagem 11.2 **results.php – Recupera resultados de pesquisa do nosso banco de dados do MySQL e os formata para exibição**

```
<html>
<head>
  <title>Book-O-Rama Search Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
  //cria nome de variável abreviado
  $searchtype=$HTTP_POST_VARS['searchtype'];
  $searchterm=$HTTP_POST_VARS['searchterm'];

  $searchterm= trim($searchterm);

  if (!$searchtype || !$searchterm)
  {
     echo 'You have not entered search details.  Please go back and try again.';
     exit;
  }

  $searchtype = addslashes($searchtype);
  $searchterm = addslashes($searchterm);

  @ $db = mysql_pconnect('localhost', 'bookorama', 'bookorama123');

  if (!$db)
  {
     echo 'Error: Could not connect to database.  Please try again later.';
     exit;
  }

  mysql_select_db('books');
  $query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
  $result = mysql_query($query);

  $num_results = mysql_num_rows($result);
 echo '<p>Number of books found: '.$num_results.'</p>';
```

Listagem 11.2 **Continuação**

```
  for ($i=0; $i <$num_results; $i++)
  {
     $row = mysql_fetch_array($result);
     echo '<p><strong>'.($i+1).'. Title: ';
     echo htmlspecialchars(stripslashes($row['title']));
     echo '</strong><br />Author: ';
     echo stripslashes($row['author']);
     echo '<br />ISBN: ';
     echo stripslashes($row['isbn']);
     echo '<br />Price: ';
     echo stripslashes($row['price']);
     echo '</p>';
  }
?>

</body>
</html>
```

Observe que o script permite que você insira os caracteres curinga de MySQL % e _ (sublinhado). Essa capacidade pode ser útil para o usuário. É possível escapar esses caracteres se forem causar problemas para sua aplicação.

A Figura 11.2 ilustra os resultados da utilização desse script para realizar uma pesquisa.

Figura 11.2 Os resultados de pesquisar livros sobre Java no banco de dados são apresentados em uma página Web utilizando o script `results.php`.

Consultando um banco de dados Web

Em qualquer script utilizado para acessar um banco de dados Web, você seguirá alguns passos básicos:

1. Verificar e filtrar dados vindos do usuário.
2. Configurar uma conexão com o banco de dados apropriado.
3. Consultar o banco de dados.
4. Recuperar os resultados.
5. Apresentar os resultados de volta para o usuário.

Esses são os passos que seguimos no script `results.php` e examinaremos cada um deles separadamente.

Verificando e filtrando dados de entrada

Começamos nosso script eliminando todos os espaços em branco que o usuário possa ter inserido inadvertidamente no início ou no final do termo de pesquisa. Fazemos isso aplicando a função `trim( )` ao `$searchterm`.

```
$searchterm=trim($searchterm);
```

Nosso próximo passo é verificar se o usuário inseriu um termo de pesquisa e um tipo de pesquisa. Observe que verificamos se ele inseriu um termo de pesquisa depois de eliminar os espaços em branco das extremidades de `$searchterm`. Se tivéssemos organizado essas linhas na ordem inversa, poderíamos ter situações em que o termo de pesquisa de um usuário não estaria vazio, assim ele não criaria uma mensagem de erro, mas conteria apenas espaços em branco, que, portanto, seriam excluídos por `trim( )`:

```
if (!$searchtype || !$searchterm)
{
   echo 'You have not entered search details.  Please go back and try again.';
   exit;
}
```

Você verificará a variável `$searchtype` mesmo que nesse caso ela venha de uma `SELECT` HTML. Talvez você pergunte por que se preocupar em verificar dados que precisam ser preenchidos. É importante lembrar que há mais de uma interface para seu banco de dados. Por exemplo, a Amazon tem muitos afiliados que utilizam sua interface de pesquisa. Além disso, é sensato verificar dados em caso de qualquer problema de segurança que possa surgir por causa de usuários que vêm de diferentes pontos de entrada.

Além disso, quando você vai utilizar qualquer entrada de dados fornecida por um usuário, é importante filtrar dela todos os caracteres de controle apropriadamente. Como você se lembra, no Capítulo 4, discutimos as funções addslashes( ), stripslashes( ) e `get_magic_quotes_gpc`. Você precisa escapar dados ao enviar qualquer entrada de usuário para um banco de dados como o MySQL.

Nesse caso, você verifica o valor da função `get_magic_quotes_gpc( )`. Ela diz se as aspas estão sendo inseridas automaticamente. Se não estiver, utilize `addslashes( )` para escapar os dados:

```
if (!get_magic_quotes_gpc( ))
{
 $searchtype = addslashes($searchtype);
 $searchterm = addslashes($searchterm);
}
```

Utilizamos também `stripslashes( )` nos dados retornados do banco de dados. Se o recurso de aspas mágicas estiver ativado, os dados terão barras quando retornarem do banco de dados, então será preciso retirá-las.

Estamos utilizando a função `htmlspecialchars( )` para codificar caracteres que têm significados especiais em HTML. Nossos dados de teste atuais não incluem os símbolos de "e" comercial (&), menor que (<), maior que (>), nem aspas duplas ("), mas muitos títulos de livro bons contêm um "e" comercial. Utilizando essa função, podemos eliminar erros futuros.

Configurando uma conexão

O PHP5 possui uma nova biblioteca para conectar-se ao MySQL, chamada mysqli (o i significa *improved*, aprimorada). A biblioteca mysqli é adequada para uso com o MySQL versão 4 e superior. Na versão 4, foi acrescentado um novo protocolo de conexão ao MySQL que é muito mais rápido, e a mysqli permite tirar vantagem dele. A biblioteca mysqli permite utilizar uma sintaxe procedural ou orientada a objetos.

Utilize a seguinte linha no script para conectar-se ao servidor MySQL:

```
@ $db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');
```

Essa linha instancia a classe `mysqli` e cria uma conexão ao host `'localhost'` com o nome de usuário `'bookorama'` e a senha `'bookorama123'`. A conexão é configurada para utilizar o banco de dados chamado books.

Utilizando essa abordagem orientada a objetos, você pode chamar métodos nesse objeto para acessar o banco de dados. A mysqli também a permite uma abordagem procedural, se você preferir. Para conectar-se de uma forma procedural, utilize

```
@ $db = mysqli_connect('localhost', 'bookorama', 'bookorama123', 'books');
```

Essa função retorna um recurso, em vez de um objeto. Esse recurso representa a conexão ao banco de dados, e se você estiver utilizando a abordagem procedural, precisará passar o recurso a todas as outras funções mysqli. Isso é semelhante à forma como funcionam as funções de manipulação de arquivo, como `fopen( )`.

A maioria das funções mysqli tem uma interface orientada a objetos e uma interface procedural. Geralmente, as diferenças são que os nomes de função na versão procedural começam com `mysqli_` e exigem que você passe o handle de recurso obtido de `mysqli_connect( )`. As conexões a bancos de dados são uma exceção a essa regra porque podem ser feitas pelo construtor do objeto mysqli.

Vale a pena verificar o resultado da tentativa de conexão porque o código restante não funcionará sem uma conexão válida de banco de dados. Para isso, utilize o seguinte código:

```
if (mysqli_connect_errno( ))
{
  echo 'Error: Could not connect to database.  Please try again later.';
  exit;
}
```

(Esse código é o mesmo para a versão orientada a objetos e a procedural.) A função `mysqli_connect_errno( )` retorna um número de erro, se houver erros, ou zero, se tiver sucesso.

Observe que quando você se conecta ao banco de dados, começa a linha do código com o operador de supressão de erro `@`. Dessa forma, é possível manipular qualquer erro de forma elegante. (Isso também pode ser feito com exceções, que não utilizamos neste exemplo.)

Lembre-se de que existe um limite ao número de conexões MySQL que podem existir ao mesmo tempo. O parâmetro MySQL `max_connections` determina qual é o limite. O objetivo desse parâmetro e do parâmetro Apache relacionado `MaxClients` é dizer ao servidor para rejeitar novos pedidos de conexão e não permitir que os recursos da máquina sejam utilizados completamente quando o sistema está ocupado ou quando o software estiver sobrecarregado.

Você pode alterar os valores default dos dois parâmetros editando os arquivos de configuração. Para definir `MaxClients` no Apache, edite o arquivo `httpd.conf` no sistema. Para definir `max_connections<` para o MySQL, edite o arquivo `my.conf`.

Escolhendo um banco de dados para utilizar

Lembre-se de que enquanto utilizamos o MySQL a partir de uma interface de linha de comando, precisamos dizer a ele qual banco de dados planejamos utilizar com um comando como:

```
use books;
```

Também precisamos fazer isso ao nos conectarmos a partir da Web. O banco de dados a ser utilizado é especificado como parâmetro ao construtor `mysqli` ou à função `mysqli_connect( )`. Se quiser mudar o banco de dados default, você pode fazê-lo com a função `mysqli_select_db( )`. Ela pode ser acessada assim

```
$db->select_db(nomedobancodedados)
```

ou

```
mysqli_select_db(recurso_banco de dados; nome_do_banco de dados)
```

Aqui, você pode ver a semelhança entre as funções que descrevemos anteriormente: a versão procedural começa com mysqli_ e requer o parâmetro handle extra do banco de dados.

Consultando o banco de dados

Para realmente realizar a consulta, podemos utilizar a função mysql_query( ). Antes de fazer isso, entretanto, é uma boa idéia configurar a consulta que desejamos executar:

```
$query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
```

Nesse caso, estamos procurando o valor inserido pelo usuário ($searchterm) no campo que o usuário especificou ($searchtype). Você notará que utilizamos like para correspondência em vez de igual – normalmente é uma boa idéia ser mais tolerante em uma pesquisa de banco de dados.

> **Dica** É importante saber que a consulta que você envia para o MySQL não precisa de um ponto-e-vírgula no final, diferente de uma consulta que você digita no monitor do MySQL.

Agora podemos executar a consulta:

```
$result = mysql_query($query);
```

Ou, se quiséssemos utilizar a interface procedural, usaríamos:

```
$result = mysqli_query($db, $query);
```

Você o passa para a consulta que deseja executar e, na interface procedural, para o link de banco de dados (novamente, nesse caso $db).

A versão orientada a objeto retorna um objeto de resultado. (Isso é parecido com o modo como as funções de conexão funcionam.) De qualquer modo, o resultado deve ser armazenado em uma variável ($result) para uso futuro. Essa função retorna false no caso de ser malsucedida.

Recuperando resultados da consulta

Há uma variedade de funções disponíveis para dividir os resultados do identificador de resultado de diferentes maneiras. O identificador de resultado é a chave para acessar zero, uma ou mais linhas retornadas pela consulta.

Nesse exemplo, contamos o número de linhas retornadas e também utilizamos a função mysqli_fetch_assoc( ).

Ao utilizarmos a abordagem orientada a objetos, o número de linhas retornadas é armazenado no membro num_rows do objeto resultante, e podemos acessá-lo assim:

```
$num_results = $result->num_rows;
```

Ao utilizar uma abordagem procedural, a função mysqli_num_rows( ) oferece o número de linhas retornadas pela consulta. Devemos passar a ela o identificador de resultados, desta forma:

```
$num_results = mysqli_num_rows($result);
```

É útil saber isso se planejamos processar ou exibir os resultados; sabemos quantas existem, podemos agora fazer um loop por elas:

```
for ($i=0; $i <$num_results; $i++)
{
  // resultados do processo
}
```

Em cada iteração desse loop, chamamos `$result->fetch_assoc( )` (ou `mysqli_fetch_assoc( )`). O loop não executa se nenhuma linha for retornada. Essa é uma função que toma cada linha do conjunto de resultados e a retorna como array, com cada chave um nome de atributo, e cada valor o valor correspondente no array:

```
$row = $result->fetch_assoc( );
```

Ou você pode usar uma abordagem procedural:

```
$row = mysqli_fetch_assoc($result);
```

Com o array `$row`, podemos percorrer cada campo e exibi-lo de forma apropriada, como mostra este exemplo:

```
echo '<br />ISBN: ';
echo stripslashes($row['isbn']);
```

Como mencionamos anteriormente, você chama `stripslashes( )` para arrumar o valor antes de exibi-lo.

Diversas variações podem ser usadas para obter os resultados de um identificador de resultados. Em vez de um array com chaves identificadas, você pode recuperar os resultados em um array enumerado com `mysqli_fetch_row( )`, da seguinte maneira:

```
$row = $result->fetch_row($result);
```

ou

```
$row = mysqli_fetch_row($result);
```

Aqui, os valores do atributo são listados em cada um dos valores de array `$row[0]`, `$row[1]` etc. (A função `mysqli_fetch_array( )` permite buscar uma linha com um ou os dois tipos de array.)

Você também poderia buscar uma linha em um objeto com a função `mysqli_fetch_object( )`:

```
$row = $result->fetch_object( );
```

ou

```
$row = mysqli_fetch_object($result);
```

Então, é possível acessar cada um dos atributos via `$row->title`, `$row->author` etc.

Desconectando-se do banco de dados

Você pode liberar seu conjunto de resultados chamando

```
$result->free( );
```

ou

```
mysqli_free_result($result);
```

Então, pode utilizar

```
$db->close( );
```

ou

```
mysqli_close($db);
```

para fechar a conexão com o banco de dados. Esse comando não é estritamente necessário porque a conexão será fechada quando o script terminar a execução de qualquer maneira.

Colocando novas informações no banco de dados

Inserir novos itens no banco de dados é notavelmente semelhante a remover itens do banco de dados. Você segue os mesmos passos básicos – estabelece uma conexão, envia uma consulta e verifica os resultados. Nesse caso, a consulta que você envia será uma `INSERT` em vez de uma `SELECT`.

Embora tudo isso seja muito parecido, às vezes é útil ver um exemplo. Na Figura 11.3, você pode ver um formulário HTML básico para colocar novos livros no banco de dados.

Figura 11.3 Essa interface para colocar novos livros no banco de dados poderia ser utilizada pelos funcionários da Book-O-Rama.

A HTML para essa página é mostrada na Listagem 11.3.

Listagem 11.3 **newbook.html** – **HTML para página de entrada de livro**

```
<html>
<head>
  <title>Book-O-Rama - New Book Entry</title>
</head>

<body>
  <h1>Book-O-Rama - New Book Entry</h1>

  <form action="insert_book.php" method="post">
    <table border="0">
      <tr>
        <td>ISBN</td>
         <td><input type="text" name="isbn" maxlength="13" size="13"><br /></td>
      </tr>
      <tr>
        <td>Author</td>
        <td> <input type="text" name="author" maxlength="30" size="30"><br /></td>
      </tr>
      <tr>
        <td>Title</td>
        <td> <input type="text" name="title" maxlength="60" size="30"><br></td>
      </tr>
      <tr>
        <td>Price $</td>
        <td><input type="text" name="price" maxlength="7" size="7"><br /></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" value="Register"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Os resultados desse formulário são passados para insert_book.php, um script que cuida de detalhes, realiza algumas validações menores e tenta escrever os dados no banco de dados. O código para esse script é mostrado na Listagem 11.4.

Listagem 11.4 **insert_book.php – Esse script grava novos livros no banco de dados**

```
<html>
<head>
  <title>Book-O-Rama Book Entry Results</title>
</head>
<body>
<h1>Book-O-Rama Book Entry Results</h1>
<?php
  //cria nome de variável abreviado
  $isbn=$HTTP_POST_VARS['isbn'];
  $author=$HTTP_POST_VARS['author'];
  $title=$HTTP_POST_VARS['title'];
  $price=$HTTP_POST_VARS['price'];

  if (!$isbn || !$author || !$title || !$price)
  {
     echo 'You have not entered all the required details.<br />'
          .'Please go back and try again.';
     exit;
  }

  $isbn = addslashes($isbn);
  $author = addslashes($author);
  $title = addslashes($title);
  $price = doubleval($price);

  @ $db = mysql_pconnect('localhost', 'bookorama', 'bookorama123');

  if (!$db)
  {
     echo 'Error: Could not connect to database.  Please try again later.';
     exit;
  }

  mysql_select_db('books');
  $query = "insert into books values
           ('".$isbn."', '".$author."', '".$title."', '".$price."')";
  $result = mysql_query($query);
  if ($result)
      echo  mysql_affected_rows( ).' book inserted into database.';
?>

</body>
</html>
```

Os resultados de inserir um livro com sucesso são mostrados na Figura 11.4.

Se examinar o código para insert_book.php, você notará que grande parte dele é semelhante ao script que escrevemos para recuperar dados do banco de dados. Verificamos que todos os campos de formulário foram preenchidos e os formatamos corretamente para inserção no banco de dados (se exigido) com addslashes( ):

```
if (!get_magic_quotes_gpc( ))
{
$isbn = addslashes($isbn);
```

```
$author = addslashes($author);
$title = addslashes($title);
$price = doubleval($price);
}
```

Figura 11.4 O script é concluído com sucesso e informa que o livro foi adicionado ao banco de dados.

Como o preço é armazenado no banco de dados como um flutuante, não queremos colocar barras nele. Podemos alcançar o mesmo efeito da filtragem de quaisquer caracteres estranhos nesse campo numérico chamando `doubleval( )`, que discutimos no Capítulo 1. Isso também cuidará de quaisquer símbolos de moeda que o usuário possa ter digitado no formulário.

Novamente, conectamo-nos ao banco de dados utilizando `mysql_pconnect( )` e configuramos uma consulta para enviar ao banco de dados. Nesse caso, a consulta é uma `INSERT` de SQL:

```
$query = "insert into books values
        ('".$isbn."', '".$author."', '".$title."', '".$price."')";
$result = mysql_query($query);
```

Essa consulta é executada no banco de dados da maneira normal chamando `$db->query( )` (ou `mysqli_query( )` se você deseja fazer as coisas de forma procedural)..

Uma diferença significativa entre utilizar `INSERT` e `SELECT` é no uso do `mysql_affected_rows( )`:

```
echo  mysql_affected_rows( )." book inserted into database.";
```

No script anterior, utilizamos `mysql_num_rows( )` para determinar quantas linhas foram retornadas por uma `SELECT`. Quando escreve consultas que alteram o banco de dados como `INSERTs`, `DELETEs` e `UPDATEs`, você deve utilizar `mysql_affected_rows( )` no lugar.

Isso abrange os princípios básicos da utilização de bancos de dados MySQL com PHP.

Utilizando instruções preparadas

A biblioteca mysqli oferece suporte ao uso de instruções preparadas. Elas são úteis para aumentar a velocidade da execução, quando você está realizando muita quantidade da mesma consulta com dados diferentes. Elas também protegem contra ataques *injection-style* no SQL.

O conceito básico de uma instrução preparada é que você envia um modelo da consulta que deseja executar ao MySQL e então envia os dados separadamente. Você pode enviar vários lotes dos mesmos dados à mesma instrução preparada; essa capacidade é particularmente útil para grandes inserções.

Você poderia utilizar instruções preparadas no script `insert_book.php` da seguinte forma:

```
$query = "insert into books values(?, ?, ?, ?)";
$stmt = $db->prepare($query);
$stmt->bind_param("sssd", $isbn, $author, $title, $price);
$stmt->execute( );
```

```
echo $stmt->affected_rows.' book inserted into database.';
$stmt->close( );
```

Vamos considerar o código linha por linha.

Ao configurar a consulta, em vez de substituir as variáveis, como era feito anteriormente, você insere pontos de interrogação para cada dado. Não coloque aspas ou outros delimitadores em volta dos pontos de interrogação.

A segunda linha é uma chamada a `$db->prepare( )`, que é `mysqli_stmt_prepare( )` na versão procedural. Essa linha constrói um objeto ou recurso da instrução que então utilizará para fazer o processamento real.

O objeto de instrução possui um método chamado `bind_param( )`. (Na versão procedural, é chamado de `mysqli_stmt_bind_param( )`.) O propósito de `bind_param( )` é informar ao PHP que variáveis devem substituir os pontos de interrogação. O primeiro parâmetro é uma string de formato, semelhante à string de formato usada em `printf( )`. O valor que você está passando aqui (`"sssd"`) significa que os quatro parâmetros são uma string, uma string, uma string e um double, respectivamente. Outros caracteres possíveis na string de formato são `i` para integer e `b` para blob. Depois desse parâmetro, é preciso listar o mesmo número de variáveis dos pontos de interrogação na instrução. Eles serão substituídos nessa ordem.

A chamada a `$stmt->execute( )` (`mysqli_stmt_execute( )` na versão procedural) na verdade executa a consulta. Você então pode acessar o número de linhas afetadas e fechar a instrução.

E, então, o quão útil é a instrução preparada? O que é inteligente é que você pode mudar os valores das quatro variáveis e reexecutar a instrução sem ter de preparar novamente. Essa capacidade é útil para realizar loops em grandes inserções.

Assim como os parâmetros, você pode vincular resultados. Para consultas do tipo `SELECT`, você pode utilizar `$stmt->bind_result( )` (ou `mysqli_stmt_bind_result( )`) para fornecer uma lista de variáveis nas quais gostaria que as colunas de resultado fossem preenchidas. Cada vez que você chamar `$stmt->fetch( )` (ou `mysqli_stmt_fetch( )`), os valores das colunas da próxima linha no conjunto de resultados serão preenchidos nessas variáveis vinculadas. Por exemplo, no script de busca de livros, que vimos anteriormente, você poderia utilizar

```
$stmt->bind_result($isbn, $author, $title, $price);
```

para vincular essas quatro variáveis às quatro colunas que serão retornadas da consulta. Depois de chamar

```
$stmt->execute( );
```

você pode chamar

```
$stmt->fetch( );
```

no loop. A cada vez que for chamado, ele busca a próxima linha do resultado nas quatro variáveis vinculadas.

Você também pode utilizar `mysqli_stmt_bind_param( )` e `mysqli_stmt_bind_result( )` no mesmo script.

Observe que na época em que este livro estava sendo escrito (PHP5RC2), as instruções preparadas geravam um bug de conflito com o Apache no Windows, mas funcionavam bem em plataformas Unix.

Usando outras interfaces de banco de dados do PHP

O PHP suporta bibliotecas para conectar-se a uma grande variedade de bancos de dados incluindo Oracle, Microsoft SQL Server e PostgreSQL.

Em geral, os princípios de conectar-se e de consultar qualquer um desses bancos de dados são quase os mesmos. Os nomes individuais de função variam e diferentes bancos de dados têm funcio-

nalidades ligeiramente diferentes, mas se você pode conectar-se ao MySQL, deve poder adaptar facilmente seu conhecimento a qualquer um dos outros.

Se quiser utilizar um banco de dados que não tenha uma biblioteca específica disponível no PHP, você pode utilizar as funções genéricas de ODBC. ODBC significa *Open Database Connectivity* e é um padrão para conexões com bancos de dados. O ODBC tem as funcionalidades mais limitadas de todos os conjuntos de função, por razões relativamente óbvias. Se tiver de ser compatível com tudo, você deve ter uma base comum e excluir recursos particulares de cada um.

Além das bibliotecas que acompanham o PHP, as classes de abstração de banco de dados que estão disponíveis, como a Metabase ou PEAR::DB, permitem utilizar os mesmos nomes de função para cada tipo diferente de banco de dados.

Utilizando uma interface genérica de banco de dados: PEAR DB

Veremos um exemplo breve que utiliza a camada de abstração PEAR DB. Isso é um dos componentes centrais do PEAR e provavelmente o mais amplamente utilizado entre todos os componentes PEAR. Se tiver o PEAR instalado, você já deverá ter o DB. Se não, consulte a seção, "Instalação do PEAR", no Apêndice A.

Para propósitos comparativos, vamos examinar como teríamos escrito nosso script dos resultados da pesquisa de maneira diferente utilizando o DB.

Listagem 11.5 **results_generic.php – Recupera resultados de pesquisa do banco de dados do MySQL e os formata para exibição**

```
<html>
<head>
  <title>Book-O-Rama Search Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
  //cria nome de variável abreviado
  $searchtype=$HTTP_POST_VARS['searchtype'];
  $searchterm=$HTTP_POST_VARS['searchterm'];

  $searchterm= trim($searchterm);

  if (!$searchtype || !$searchterm)
  {
     echo 'You have not entered search details.  Please go back and try again.';
     exit;
  }
 $searchtype = addslashes($searchtype);
  $searchterm = addslashes($searchterm);

  // configura para utilizar PEAR DB
  require('DB.php');
  $user = 'bookorama';
  $pass = 'bookorama123';
  $host = 'localhost';
  $db_name = 'books';

  // configura a string de conexão universal ou DSN
  $dsn = "mysql://$user:$pass@$host/$db_name";

  // conecta-se ao banco de dados
  $db = DB::connect($dsn, true);
```

Listagem 11.5 **Continuação**

```
  // verifica se a conexão funcionou
  if (DB::isError($db))
  {
    echo $db->getMessage( );
    exit;
  }

  // realiza uma consulta
  $query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
  $result = $db->query($query);
  // verifica se o resultado foi ok
  if (DB::isError($result))
  {
    echo $db->getMessage( );
    exit;
  }

  // obtém o número de linhas retornadas
  $num_results = $result->numRows( );

  // exibe cada linha retornada
  for ($i=0; $i <$num_results; $i++)
  {
     $row = $result->fetchRow(DB_FETCHMODE_ASSOC);
     echo '<p><strong>'.($i+1).'. Title: ';
     echo htmlspecialchars(stripslashes($row['title']));
     echo '</strong><br />Author: ';
     echo stripslashes($row['author']);
     echo '<br />ISBN: ';
     echo stripslashes($row['isbn']);
     echo '<br />Price: ';
     echo stripslashes($row['price']);
     echo '</p>';
  }

  // desconecta do banco de dados
  $db->disconnect( );
?>

</body>
</html>
```

Vamos examinar o que estamos fazendo de diferente nesse script.

Para nos conectar ao banco de dados utilizamos a linha:

```
$db = DB::connect($dsn);
```

Essa função aceita uma string de conexão universal que contém todos os parâmetros necessários para uma conexão com o banco de dados. Você pode ver isso se examinar o formato da string de conexão:

```
$dsn = "mysql://$user:$pass@$host/$db_name";
```

Depois disso, verificamos se a conexão foi malsucedida utilizando o método `isError( )` e, se foi, imprimimos a mensagem de erro e encerramos:

```
if (DB::isError($db))
{
  echo $db->getMessage( );
```

```
  exit;
}
```

Assumindo que tudo deu certo, em seguida configuramos e executamos uma consulta da seguinte maneira:

```
$result = $db->query($query);
```

Podemos verificar o número de linhas retornadas:

```
$num_results = $result->numRows( );
```

Recuperamos cada linha da seguinte maneira:

```
$row = $result->fetchRow(DB_FETCHMODE_ASSOC);
```

O método `fetchRow( )` genérico pode buscar uma linha em muitos formatos diferentes – o parâmetro `DB_FETCHMODE_ASSOC` informa que gostaríamos que a linha retornasse como um array associativo.

Depois de dar saída para as linhas retornadas, concluímos fechando a conexão do banco de dados:

```
$db->disconnect( );
```

Como você pode ver, esse exemplo genérico é muito semelhante ao nosso primeiro script.

As vantagens da utilização do DB são o fato de que precisamos lembrar de apenas um conjunto de funções do banco de dados e que o código exigirá alterações mínimas se decidirmos alterar nosso software do banco de dados.

Como este é um livro sobre MySQL utilizaremos as bibliotecas nativas do MySQL para um pouco mais de velocidade e flexibilidade. Mas você poderia querer utilizar o pacote DB nos seus projetos, uma vez que há momentos em que o uso de uma camada de abstração pode ser extremamente útil.

Leitura adicional

Para obter informações adicionais sobre como combinar MySQL e PHP, você pode ler as seções apropriadas nos manuais do PHP e do MySQL.

Para obter mais informações sobre o ODBC, visite:

http://www.webopedia.com/TERM/O/ODBC.html

A seguir

No próximo capítulo, entraremos em mais detalhe sobre administração do MySQL e discutiremos como você pode otimizar seus bancos de dados.

12

MySQL avançado

NESTE CAPÍTULO, ABORDAREMOS ALGUNS TÓPICOS mais avançados do MySQL incluindo privilégios avançados, segurança e otimização.

Os tópicos que abordaremos são:

- Entendendo o sistema de privilégios em detalhe;
- Tornando o banco de dados MySQL seguro;
- Obtendo informações adicionais sobre bancos de dados;
- Acelerando com índices;
- Otimizando sei banco de dados;
- Backup e recuperação;
- Implementando replicação

Entendendo o sistema de privilégios em detalhe

Anteriormente (no Capítulo 9), vimos como configurar usuários e conceder-lhes privilégios. Fizemos isso com o comando `GRANT`. Se você vai administrar um banco de dados MySQL, pode ser útil entender exatamente o que `GRANT` faz e como ele funciona.

Quando você emite uma instrução `GRANT`, ela afeta tabelas no banco de dados especial chamado `mysql`. As informações de privilégio são armazenadas em cinco tabelas nesse banco de dados. Por isso, ao conceder privilégios sobre os bancos de dados, você deve ser cauteloso ao conceder acesso ao banco de dados `mysql`.

Uma nota paralela é que o comando `GRANT` só está disponível a partir da versão do MySQL 3.22.11 avançada.

Podemos ver o que há dentro do banco de dados `mysql` efetuando logon como um administrador e digitando:

```
use mysql;
```

Se fizer isso, você então visualiza as tabelas nesse banco de dados digitando

```
show tables;
```

como de costume.

Os resultados que você obtém serão algo assim:

```
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| help_category   |
| help_keyword    |
| help_relation   |
| help_topic      |
| host            |
| proc            |
| tables_priv     |
| user            |
+-----------------+
```

Cada uma dessas tabelas armazena informações do sistema. Cinco delas – `user`, `host`, `db`, `tables_priv` e `columns_priv` – armazenam informações de privilégio. Às vezes elas são *tabelas grant*. Essas tabelas variam em sua função específica, mas servem todas a mesma função geral, que é determinar o que os usuários têm ou não permissão de fazer. Cada uma delas contém dois tipos de campos: campos de escopo, que identificam o usuário, host e parte de um banco de dados ao qual o privilégio se refere; e campos de privilégio, que identificam quais ações podem ser realizadas pelo usuário naquele escopo.

As tabelas `user` e `host` são utilizadas para decidir se um usuário pode se conectar ao servidor MySQL e se possui qualquer privilégio de administrador. As tabelas `db` e `host` determinam que bancos de dados o usuário pode acessar. A tabela `tables_priv` determina que tabelas dentro de um banco de dados o usuário pode utilizar, e a tabela `columns_priv` determina a que colunas dentro das tabelas o usuário tem acesso.

A tabela user

Essa tabela contém os detalhes dos privilégios do usuário global, determina se um usuário tem permissão para se conectar ao servidor do MySQL e se ele tem algum privilégio de nível global; isto é, privilégios que se aplicam a cada banco de dados no sistema.

Podemos ver a estrutura dessa tabela emitindo uma instrução `describe user;`.

O esquema para a tabela user é mostrado na Tabela 12.1.

Tabela 12.1 **Esquema da tabela user no banco de dados mysql**

Campo	Tipo
Host	varchar(60)
User	varchar(16)
Password	varchar(41)
Select_priv	enum('N','Y')
Insert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Reload_priv	enum('N','Y')

Tabela 12.1 Continuação

Campo	Tipo
Shutdown_priv	enum('N','Y')
Process_priv	enum('N','Y')
File_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
Index_priv	enum('N','Y')
Alter_priv	enum('N','Y')
Show_db_priv	enum('N','Y')
Super_priv	enum('N','Y')
Create_tmp_table_priv	enum('N','Y')
Lock_tables_priv	enum('N','Y')
Execute_priv	enum('N','Y')
Repl_slave_priv	enum('N','Y')
Repl_client_priv	enum('N','Y')
ssl_type	enum('N','Y')
ssl_cipher	enum('N','Y')
x509_issuer	enum('N','Y')
x509_subject	enum('N','Y')
max_questions	enum('N','Y')
max_updates	enum('N','Y')
max_connections	enum('N','Y')

Cada linha nessa tabela corresponde a um conjunto de privilégios para um usuário que vem de um host e efetuando logon com a senha Password. Esses são os *campos de escopo* para essa tabela, uma vez que eles descrevem o escopo dos outros campos, chamados *campos de privilégio.*

Os privilégios listados nessa tabela (e os outros a seguir) correspondem aos privilégios que concedemos utilizando GRANT no Capítulo 9. Por exemplo, Select_priv corresponde ao privilégio para executar um comando SELECT.

Se um usuário tiver um privilégio particular, o valor nessa coluna será Y. Inversamente, se um usuário não tiver recebido esse privilégio, o valor será N.

Todos os privilégios listados na tabela user são globais, isto é, se aplicam a *todos os bancos de dados no sistema* (incluindo o banco de dados mysql). Portanto, os administradores terão alguns Ys aí, mas a maioria dos usuários deverá ter só Ns. Os usuários normais devem ter direitos sobre os bancos de dados apropriados, não sobre todas as tabelas.

Tabelas db e host

A maioria dos privilégios dos usuários médios é armazenada nas tabelas db e host.

A tabela db determina quais usuários podem acessar quais bancos de dados a partir de quais hosts. Os privilégios listados nessa tabela se aplicam a qualquer banco de dados que seja chamado em uma linha particular.

A tabela host suplementa a tabela db. Se um usuário estiver conectado a um banco de dados de múltiplos hosts, nenhum host será listado para esse usuário na tabela db. Em vez disso, o usuário terá um conjunto de entradas na tabela host, uma para cada combinação usuário-host, especificando seus privilégios.

Os esquemas dessas duas tabelas são mostrados na Tabelas 12.2 e 12.3, respectivamente.

Tabela 12.2 **Esquema da tabela Db no banco de dados mysql**

Campo	Tipo
Host	char(60)
Db	char(64)
User	char(16)
Select_priv	enum('N','Y')
Insert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
Index_priv	enum('N','Y')
Alter_priv	enum('N','Y')
Create_tmp_tables_priv	enum('N','Y')
Lock_tables_priv	enum('N','Y')

Tabela 12.3 **Esquema da tabela host no banco de dados mysql**

Campo	Tipo
Host	char(60)
Db	char(64)
Select_priv	enum('N','Y')
Insert_priv	enum('N','Y')
Update_priv	enum('N','Y')
Delete_priv	enum('N','Y')
Create_priv	enum('N','Y')
Drop_priv	enum('N','Y')
Grant_priv	enum('N','Y')
References_priv	enum('N','Y')
Index_priv	enum('N','Y')
Alter_priv	enum ('N','Y')
Create_tmp_tables_priv	enum('N','Y')
Lock_tables_priv	enum('N','Y')

Tabelas tables_priv e columns_priv

Essas duas tabelas são utilizadas para armazenar privilégios nos níveis de tabela e de coluna, respectivamente. Elas funcionam como a tabela db, exceto que fornecem privilégios sobre tabelas dentro de um banco de dados específico e colunas dentro de uma tabela específica, respectivamente.

Essas tabelas têm uma estrutura ligeiramente diferente para as tabelas user, db e host. Os esquemas para as tabelas tables_priv e columns_priv são mostrados nas Tabelas 12.4 e 12.5, respectivamente.

Tabela 12.4 **Esquema da tabela tables_priv no banco de dados mysql**

Campo	Tipo
Host	char(60)
Db	char(64)
User	char(16)
Table_name	char(60)
Grantor	char(77)
Timestamp	timestamp(14)
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter')
Column_pri	set ('Select', 'Insert', 'Update', 'References')

Tabela 12.5 **Esquema da tabela columns_priv no banco de dados mysql**

Campo	Tipo
Host	char(60)
Db	char(64)
User	char(16)
Table_name	char(64)
Column_name	char(64)
Timestamp	timestamp(14)
Column_priv	set('Select', 'Insert', 'Update, 'Reference')

A coluna Grantor na tabela tables_priv armazena o usuário que concedeu esse privilégio a esse usuário. A coluna Timestamp nessas duas tabelas armazena a data e a hora em que o privilégio foi concedido.

Controle de acesso: como o MySQL utiliza as tabelas grant

O MySQL utiliza as tabelas grant para determinar o que um usuário tem permissão de fazer em um processo de duas etapas:

1. Verificação de conexão. Aqui, o MySQL verifica se você tem ou não permissão de conectar-se, baseado nas informações da tabela de usuário, como mostrado anteriormente. Isso é baseado em seu nome de usuário, hostname e senha. Se um nome de usuário estiver em branco, ele faz correspondência com todos os usuários. Hostnames podem ser especificados com

um curinga (%). Isso pode ser utilizado como o campo inteiro – ou seja, % corresponde a todos os hosts – ou como parte de um nome de host, por exemplo, %.tangledweb.com.au corresponde a todos os hosts que terminam em .tangledweb.com.au. Se o campo de senha estiver em branco, nenhuma senha será exigida. É mais seguro evitar ter usuários em branco, curingas em hosts e usuários sem senhas. Se o hostname estiver em branco, MySQL refere-se à tabela host para uma entrada user e host correspondente.

2. Verificação de solicitação. Toda vez que inserir uma solicitação, depois que estabeleceu uma conexão, o MySQL verifica se você tem o nível apropriado de privilégios para realizar essa solicitação. O sistema começa verificando seus privilégios globais (na tabela user) e se eles não forem suficientes, verifica as tabelas db e host. Se você ainda não tiver privilégios suficientes, o MySQL verificará a tabela tables_priv, e, se isso não for suficiente, por fim ele verificará a tabela columns_priv.

Atualizando privilégios: quando as alterações realmente entram em vigor?

O servidor do MySQL lê automaticamente as tabelas grant quando ele é iniciado e quando você emite as instruções GRANT e REVOKE. Entretanto, agora que sabe onde e como esses privilégios são armazenados, você pode alterá-los manualmente. Quando você os atualiza manualmente, o servidor do MySQL não *notará que eles foram alterados*.

Você precisa indicar para o servidor que uma alteração ocorreu, e há três maneiras de fazer isso. Você pode digitar

```
flush privileges;
```

no prompt do MySQL (você precisará estar conectado como administrador para fazer isso). Essa é a maneira mais comumente utilizada de atualizar os privilégios.

Alternativamente, você pode executar tanto

```
mysqladmin flush-privileges
```

como

```
mysqladmin reload
```

a partir do sistema operacional.

Depois disso, os privilégios de nível global serão verificados da próxima vez que um usuário se conectar; os privilégios de banco de dados serão verificados quando a próxima instrução use for emitida; e os privilégios de nível de tabela e coluna serão verificados na próxima solicitação de um usuário.

Tornando o banco de dados MySQL seguro

A segurança é importante, especialmente quando você começar a conectar o banco de dados MySQL ao Web site. Nesta seção, veremos as precauções que você deve tomar para proteger o banco de dados.

MySQL do ponto de vista do sistema operacional

É uma idéia ruim executar o servidor do MySQL (mysqld) como root se você estiver executando um sistema operacional do tipo UNIX. Isso fornece a um usuário do MySQL com um conjunto completo de privilégios o direito de ler e gravar arquivos em qualquer lugar no sistema operacional. Esse é um ponto importante, facilmente negligenciado, que ficou famoso pelo fato de um hacker tê-lo explorado para invadir o Web site da Apache. (Felizmente, os crackers eram "do bem" e o único resultado da ação deles foi o reforço da segurança por parte da empresa.)

É uma boa idéia configurar um usuário de MySQL especificamente para o propósito de execução de mysqld. Além disso, você pode tornar os diretórios (onde os dados físicos estão armazenados) acessíveis somente pelo usuário do MySQL. Em muitas instalações, o servidor é configurado para executar como userid mysql, no grupo mysql.

Você também deve, de modo ideal, configurar seu servidor MySQL por atrás do firewall. Dessa maneira você pode parar conexões de máquinas não-autorizadas – verifique e veja se você pode conectar-se de fora para o servidor no número de porta 3306. Essa é a porta padrão em que o MySQL executa e deve estar fechada no firewall.

Senhas

Certifique-se de que todos os seus usuários tenham senhas (especialmente root!) e de que essas sejam bem escolhidas e regularmente alteradas, como com senhas de sistema operacional. A regra básica para lembrar aqui é de que senhas que são ou que contêm palavras de um dicionário são uma idéia ruim. Combinações de letras e números são melhores.

Se você vai armazenar senhas em arquivos de script, então certifique-se de que somente o usuário cuja senha está armazenada possa ver esse script.

Os scripts do PHP que são usados para se conectar ao banco de dados precisam de acesso à senha daquele usuário. Isso pode ser feito de uma forma razoavelmente segura inserindo o login e a senha em um arquivo chamado, por exemplo, dbconnect.php, que, então, você inclui quando necessário. Esse script pode ser cuidadosamente armazenado fora da árvore de documentos Web e tornado acessível apenas pelo usuário apropriado.

Lembre-se de que se você inserir esses detalhes em um arquivo com .inc ou alguma outra extensão na árvore Web, deve ter cuidado de verificar se o servidor Web sabe que esses arquivos devem ser interpretados como PHP, de modo que os detalhes não possam ser visualizados em um navegador Web.

Não armazene senhas em texto simples no banco de dados. As senhas do MySQL não são armazenadas dessa forma, mas é comum em aplicações Web você querer armazenar nomes de login e senhas de membros do Web site. É possível encriptar as senhas (de uma via) utilizando a função SHA1( ) do MySQL. Lembre-se de que se inserir uma senha nesse formato ao executar SELECT (para fazer o login de um usuário), você precisará utilizar a mesma função novamente para verificar a senha que o usuário digitou.

Utilizaremos essa funcionalidade quando formos implementar os projetos na Parte V.

Privilégios de usuário

Saber é poder. Certifique-se de que você entende o sistema de privilégios do MySQL e as conseqüências de conceder privilégios particulares. Não conceda a qualquer usuário mais privilégios do que ele precisa. Você deve verificar isso examinando as tabelas de concessão de privilégios.

Em particular, não conceda os privilégios PROCESS, FILE, SHUTDOWN e RELOAD a qualquer usuário diferente de um administrador a menos que seja absolutamente necessário. O privilégio PROCESS pode ser utilizado para ver o que outros usuários estão fazendo e digitando, incluindo suas senhas. O privilégio FILE pode ser utilizado para ler e gravar arquivos para e a partir do sistema operacional (incluindo, digamos, /etc/password em um sistema Unix).

O privilégio GRANT também deve ser concedido com cautela já que esse privilégio permite que os usuários compartilhem seus privilégios com outros.

Certifique-se de que ao configurar usuários, você só lhes conceda acesso quando proveniente dos hosts a partir dos quais eles estarão se conectando. Se você tiver jane@localhost como um usuário, tudo bem, mas apenas jane é muito comum e poderia efetuar o login de qualquer lugar – e ela pode não ser a jane que você pensa que é. Evite usar curingas em nome de hosts por razões semelhantes.

Você pode aumentar ainda mais a segurança utilizando IP em vez de nomes de domínio na sua tabela host. Isso evita problemas com erros ou crackers em seu DNS. Você pode impor isso inician-

do o daemon do MySQL com a opção `--skip-name-resolve`, o que significa que todos os valores de coluna de host devem ser endereços de IP ou host local.

Você também deve evitar que os usuários não-administrativos tenham acesso ao programa `mysqladmin` no servidor Web. Como esse programa executa a partir da linha de comando, trata-se de privilégio de sistema operacional.

Questões relacionadas à Web

Quando você conecta seu banco de dados MySQL à Web, surgem algumas questões de segurança especiais.

Não é uma idéia ruim começar configurando um usuário especial apenas para o propósito de conexões Web. Dessa maneira, você pode oferecer aos usuários o mínimo privilégio necessário e não conceder, por exemplo, privilégios `DROP`, `ALTER` ou `CREATE` a esses usuários. Talvez você conceda `SELECT` apenas sobre as tabelas de catálogo, e `INSERT` apenas sobre as tabelas de pedido. Novamente, isso é uma ilustração de como utilizar o princípio do menor privilégio.

> **Cuidado** No último capítulo, falamos sobre como utilizar as funções `addslashes( )` e `stripslashes( )` do PHP para livrar-se de qualquer caractere problemático nas strings. É importante lembrar-se de fazer isso e fazer uma limpeza geral dos dados antes de enviar qualquer coisa para o MySQL. Talvez você lembre-se de que utilizamos a função `doubleval( )` para verificar se os dados numéricos eram realmente numéricos. É um erro comum esquecer-se disso – as pessoas lembram de utilizar `addslashes( )` mas não de verificar os dados numéricos.

Você deve sempre verificar todos os dados vindos de um usuário. Mesmo que seu formulário HTML consista em caixas de seleção e botões de opção, alguém talvez altere o URL para tentar quebrar seu script. Vale a pena também verificar o tamanho dos dados recebidos.

Se os usuários estiverem digitando senhas ou dados confidenciais para serem armazenados no banco de dados, lembre-se de que os dados serão transmitidos do navegador para o servidor em texto simples a menos que você utilize a SSL (Secure Sockets Layer). Discutiremos como utilizar a SSL em mais detalhe mais tarde.

Obtendo mais informações sobre bancos de dados

Até agora, utilizamos `SHOW` e `DESCRIBE` para descobrir que tabelas estão no banco de dados e quais colunas estão nas tabelas. Veremos brevemente de que outra forma elas podem ser utilizadas e o uso da instrução `EXPLAIN` para obter informações adicionais sobre como uma `SELECT` é realizada.

Obtendo informações adicionais com `SHOW`

Anteriormente utilizamos

```
show tables;
```

para obter uma lista de tabelas no banco de dados.

A instrução

```
show databases;
```

exibirá uma lista de bancos de dados disponíveis. Você então pode utilizar a instrução `SHOW TABLES` para ver uma lista de tabelas em um desses bancos de dados:

```
show tables from books;
```

Quando você utiliza `SHOW TABLES` sem especificar um banco de dados, o que estiver em utilização é assumido por padrão.

Quando você sabe quais são as tabelas, pode obter uma lista das colunas:

```
show columns from orders from books;
```

Se você deixar o parâmetro de banco de dados desativado, a instrução SHOW COLUMNS assumirá o banco de dados atualmente em utilização por padrão. Você também pode utilizar a notação *table.column*:

```
show columns from books.orders;
```

Uma outra variação útil da instrução SHOW pode ser usada para ver que privilégios o usuário possui. Por exemplo, se executar

```
show grants for bookorama;
```

você terá a seguinte saída:

```
+---------------------------------------------------------------------+
| Grants for bookorama@%                                              |
+---------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'bookorama'@'%'                               |
| IDENTIFIED BY PASSWORD '*1ECE648641438A28E1910D0D7403C5EE9E8B0A85'  |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER    |
| ON `books`.* TO 'bookorama'@'%'                                     |
+---------------------------------------------------------------------+
```

As instruções GRANT mostradas não são necessariamente aquelas que foram executadas para fornecer privilégios àquele usuário específico, e sim instruções equivalentes resumidas que produziriam o nível atual de privilégio do usuário.

Nota A instrução SHOW GRANTS foi adicionada na versão 3.23.4 do MySQL – se você tiver uma versão anterior, essa instrução não funcionará.

Há muitas outras variações da instrução SHOW. Um resumo de todas as variações é mostrado na Tabela 12.6.

Obtendo informações sobre colunas com DESCRIBE

Como uma alternativa à instrução SHOW COLUMNS, você pode utilizar a instrução DESCRIBE, semelhante à instrução DESCRIBE no Oracle (outro RDBMS). A sintaxe básica para ela é:

```
DESCRIBE tabela [coluna ];
```

Isso fornecerá informações sobre todas as colunas na tabela ou uma coluna específica se *coluna* for especificada. Você pode utilizar curingas no nome de coluna se quiser.

Entendendo como as consultas funcionam com EXPLAIN

A instrução EXPLAIN pode ser utilizada de duas maneiras. Primeiro, você pode utilizar:

```
EXPLAIN tabela;
```

Isso fornece saída muito semelhante a DESCRIBE *tabela* ou SHOW COLUMNS FROM *tabela*.

A segunda e mais interessante maneira de utilizar EXPLAIN permite ver exatamente como o MySQL avalia uma consulta SELECT. Para utilizá-la dessa maneira, simplesmente coloque a palavra EXPLAIN na frente de uma instrução SELECT.

Tabela 12.6 **Sintaxe da instrução SHOW**

Variação	Descrição
SHOW DATABASES [LIKE *banco_de_dados*]	Lista bancos de dados disponíveis, opcionalmente com nomes do tipo banco_de_dados.
SHOW [OPEN] TABLES [FROM *banco_de_dados*] [LIKE *tabela*]	Lista tabelas do banco de dados em uso atualmente, ou do banco de dados chamado *banco_de_dados* se especificado, opcionalmente com nomes de tabelas como *tabela*.
SHOW COLUMNS FROM table [FROM *banco_de_dados*] [LIKE coluna]	Lista todas as colunas em uma tabela específica do banco de dados atualmente em uso, ou do banco de dados especificado, opcionalmente com nomes de colunas do tipo coluna. Você pode usar SHOW FIELDS em vez de SHOW COLUMNS.
SHOW INDEX FROM *tabela* [FROM *banco de dados*]	Mostra detalhes de todos os índices em uma tabela específica do banco de dados atualmente em uso, ou do banco de dados chamado *banco_de_dados* se especificado. Em vez disso, você pode usar SHOW KEYS.
SHOW STATUS [LIKE *item_do_status*]	Oferece informações de vários outros itens do sistema, como o número de threads sendo executados. A cláusula LIKE é usada para fazer correspondências com os nomes desses itens, então, por exemplo, 'Thread%' corresponde aos itens 'Threads_cached', 'Threads_connected' e 'Threads_running'.
SHOW [GLOBAL|SESSION] VARIABLES [LIKE *nome_da_variável*]	Exibe os nomes e valores das variáveis do sistema MySQL, como o número da versão. A cláusula LIKE pode ser usada para fazer correspondência de uma forma semelhante a SHOW STATUS.
SHOW [FULL] PROCESSLIST	Exibe todos os processos em execução no sistema – ou seja, as consultas que estão sendo atualmente executadas. A maioria dos usuários verá seus próprios threads, mas se eles tiverem privilégio PROCESS, podem ver os processos de todos – incluindo senhas se estiverem em consultas. As consultas são truncadas para 100 caracteres por padrão. Utilizar a palavra-chave opcional FULL exibe as consultas completas.
SHOW TABLE STATUS [FROM *banco_de_dados*] [LIKE *banco_de_dados*]	Exibe informações sobre cada uma das tabelas no banco de dados atualmente em uso, ou do banco de dados chamado *banco_de_dados* especificado, opcionalmente com uma correspondência de curinga. Essa informação inclui o tipo de tabela e a hora em que cada tabela foi atualizada pela última vez.
SHOW GRANTS FOR *usuário*	Mostra as instruções GRANT necessárias para fornecer ao usuário especificado seu nível atual de privilégio.
SHOW PRIVILEGES	Mostra os diferentes privilégios que o servidor suporta.
SHOW CREATE DATABASE *db*	Mostra uma instrução CREATE DATABASE que cria o banco de dados especificado.
SHOW CREATE TABLE *nome_da_tabela*	Mostra uma instrução CREATE TABLE que cria a tabela especificada.
SHOW [STORAGE] ENGINES	Mostra os mecanismos de armazenamento que estão disponíveis nesta instalação e que são default. (Discutimos mecanismos de armazenamento posteriormente no Capítulo 13)
SHOW INNODB STATUS	Mostra os dados sobre o estado atual do mecanismo de armazenamento InnoDB.
SHOW [BDB] LOGS	Mostra informações sobre arquivos de log para o mecanismo de armazenamento BDB.
SHOW WARNINGS [LIMIT [*deslocamento*,] *contagem_de_linha*]	Mostra erros, avisos ou observações geradas pela última instrução executada.
SHOW ERRORS [LIMIT [*deslocamento*,] *contagem_da_linha*]	Mostra apenas os erros gerados pela última instrução executada.

Você pode utilizar a instrução EXPLAIN quando estiver tentando fazer uma consulta complexa funcionar e claramente não obteve a resposta correta, ou quando uma consulta estiver levando muito mais tempo para processar do que deveria. Se estiver escrevendo uma consulta complexa, você pode verificar isso antecipadamente executando o comando EXPLAIN antes de realmente executar a consulta. Com a saída dessa instrução, você pode retrabalhar o SQL para otimizá-lo se necessário. Além disso, ela é uma ferramenta de aprendizagem útil.

Por exemplo, tente executar a seguinte consulta no banco de dados Book-O-Rama:

```
explain
select customers.name
from customers, orders, order_items, books
where customers.customerid = orders.customerid
and orders.orderid = order_items.orderid
and order_items.isbn = books.isbn
and books.title like '%Java%';
```

Essa consulta produz a seguinte saída. (Observe que estamos exibindo-a verticalmente porque as linhas da tabela são muito largas para caber neste livro. Você pode obter esse formato terminando a consulta com \G em vez do ponto-e-vírgula.)

```
*************************** 1. row ***************************
         id: 1
 select_type: SIMPLE
       table: books
        type: ALL
possible_keys: PRIMARY
         key: NULL
     key_len: NULL
         ref: NULL
        rows: 4
       Extra: Using where
*************************** 2. row ***************************
         id: 1
 select_type: SIMPLE
       table: order_items
        type: index
possible_keys: PRIMARY
         key: PRIMARY
     key_len: 17
         ref: NULL
        rows: 4
       Extra: Using where; Using index
*************************** 3. row ***************************
         id: 1
 select_type: SIMPLE
       table: orders
        type: eq_ref
possible_keys: PRIMARY
         key: PRIMARY
     key_len: 4
         ref: books.order_items.orderid
        rows: 1
       Extra:
*************************** 4. row ***************************
         id: 1
 select_type: SIMPLE
       table: customers
        type: eq_ref
possible_keys: PRIMARY
         key: PRIMARY
     key_len: 4
         ref: books.orders.customerid
        rows: 1
       Extra:
```

Essa saída pode parecer confusa, a princípio, mas pode ser muito útil. Vamos observar as colunas nessa tabela, uma a uma.

A primeira coluna, `id`, fornece o número ID da instrução `SELECT` dentro da consulta à qual essa linha se refere.

A coluna `select_type` explica o tipo de consulta sendo utilizado. O conjunto de valores que essa coluna pode ter é mostrado na Tabela 12.7.

Tabela 12.7 **Tipos possíveis de seleção, conforme mostrado na saída de `Explain`**

Tipo	Descrição
SIMPLE	O antigo SELECT, como nesse exemplo.
PRIMARY	Consulta externa (primeira) em que as subconsultas e uniões são utilizadas.
UNION	Segunda consulta ou posterior em uma união.
DEPENDENT UNION	Segunda consulta ou posterior em uma união, depende da consulta primária.
SUBQUERY	Subconsulta interna.
DEPENDENT SUBQUERY	Subconsulta interna; depende da consulta primária (ou seja, uma subconsulta correlata).
DERIVED	Subconsulta utilizada na cláusula FROM.

A coluna `table` apenas lista as tabelas utilizadas para responder à consulta. Cada linha no resultado oferece mais informações sobre como a tabela específica é utilizada nessa consulta. Nesse caso, você pode ver que as tabelas utilizadas são `orders`, `order_items`, `customers` e `books`. (Você já sabe disso observando a consulta.)

A coluna `type` explica como a tabela está sendo utilizada em junções na consulta. O conjunto de valores que essa coluna pode ter é mostrado na Tabela 12.8. Eles são listados na ordem da mais rápida para a mais lenta em termos de execução de consulta. A tabela oferece uma idéia de quantas linhas precisam ser lidas em cada tabela para executar uma consulta.

Tabela 12.8 **Possíveis tipos de junções, conforme mostrado na saída de `EXPLAIN`**

Tipo	Descrição
const ou system	A tabela é lida apenas uma vez. Isso acontece quando a tabela tem exatamente uma linha. O tipo system é utilizado quando é uma tabela de sistema e o tipo const, caso contrário.
eq_ref	Para cada conjunto de linhas de outras tabelas na junção, você lê apenas uma linha dessa tabela. Esse tipo é utilizado quando a junção usa todas as partes do índice na tabela, e o índice é UNIQUE ou a chave primária.
ref	Para cada conjunto de linhas das outras tabelas na junção, você lê um conjunto das linhas da tabela que correspondam. Esse tipo é utilizado quando a junção não pode escolher uma única linha baseada na condição da junção – ou seja, quando apenas parte da chave é utilizada na junção, ou se ela não for UNIQUE ou chave primária.
ref_or_null	É como uma consulta ref, mas o MySQL também procura por linhas que são NULL. (Esse tipo é usado principalmente em subconsultas.)
index_merge	Uma otimização específica, Index Merge, foi usada.
unique_subquery	Esse tipo de junção é usado para substituir ref em algumas subconsultas IN, em que uma única linha é retornada.

Tabela 12.8 **Continuação**

Tipo	Descrição
index_subquery	Esse tipo de junção é semelhante a unique_subquery mas é usado para suconsultas com índices não-únicos.
range	Para cada cojunto de linhas de outras tabelas na junção, você lê um conjunto de linhas de tabela que estão em um intervalo específico.
index	O índice inteiro é varrido.
ALL	Cada linha na tabela é varrida.

No exemplo anterior, você pode ver que duas das tabelas são unidas utilizando eq_ref (orders e customers), uma utilizando index (order_items) e a outra (books) é unida utilizando ALL – ou seja, examinando cada linha na tabela.

A coluna rows confirma isso: ela lista (toscamente) o número de linhas de cada tabela que deve ser varrido para realizar a junção. Você pode multiplicar esses números para obter o número total de linhas examinadas quando uma consulta é realizada. Isso porque uma junção é como um produto de linhas em diferentes tabelas. Veja o Capítulo 10 para obter mais detalhes. Lembre-se de que esse é o número de linhas examinadas, não o número de linhas retornadas, e é apenas uma estimativa. O MySQL não pode saber o número exato sem realizar a consulta.

É óbvio que quanto menor for esse número, melhor. Atualmente, você tem uma pequena quantidade de dados no banco de dados, mas quando o banco de dados começar a crescer, essa consulta pode aumentar em tempo de execução. Retornaremos a esse assunto em breve.

A coluna possible_keys lista, como você pode esperar, as chaves que o MySQL pode utilizar para unir a tabela. Nesse caso, você pode ver que as chaves possíveis são todas PRIMARY.

A coluna key é a chave da tabela MySQL realmente utilizada ou NULL se nenhuma chave tiver sido usada. Observe que, embora haja uma possível chave PRIMARY para a tabela books, ela não foi utilizada nessa consulta.

A coluna key_len indica o tamanho da chave utilizada. Você pode usar esse número para informar se apenas parte de uma chave foi utilizada. O tamanho da chave é relevante quando você tem chaves que consistem em mais de uma coluna. Nesse caso, a chave completa foi utilizada.

A coluna ref mostra as colunas utilizadas com a chave para selecionar linhas da tabela.

E, finalmente, a coluna Extra oferece qualquer outra informação sobre como a junção foi realizada. Os possíveis valores dessa coluna são mostrados na Tabela 12.9.

Tabela 12.9 **Possíveis valores para a coluna Extra, conforme mostrado em EXPLAIN**

Valor	Significado
Distinct	Depois que a primeira correspondência foi encontrada, o MySQL pára de tentar encontrar as linhas.
Not exists	A consulta foi otimizada para usar LEFT JOIN.
Range checked for each record	Para cada linha no conjunto de linhas das outras tabelas na junção, o MySQL tenta encontrar o melhor índice a ser utilizado, se houver.
Using filesort	São necessárias duas passagens para classificar os dados. (A operação obviamente demora duas vezes mais.)
Using index	Todas as informações da tabela vêm do índice; ou seja, as linhas na verdade não são pesquisadas.
Using temporary	Uma tabela temporária precisa ser criada para executar essa consulta.
Using where	Uma cláusula WHERE é utilizada para selecionar linhas.

Há várias maneiras de resolver os problemas encontrados na saída de EXPLAIN. Primeiro, você pode verificar os tipos de colunas e certificar-se de que são iguais. Isso se aplica especialmente a larguras de colunas. Os índices não podem ser utilizados para corresponder colunas se tiverem larguras diferentes. É possível resolver esse problema mudando os tipos de colunas a serem correspondidas ou construindo isso no seu próprio projeto desde o início.

Segundo, você pode dizer ao otimizador da junção para examinar as distribuições de chaves e, portanto, otimizar junções de forma mais eficiente com o utilitário myisamchk ou a instrução ANALYZE TABLE, que são equivalentes. Você pode chamar esse utilitário digitando:

```
myisamchk -analyze pathtomysqldatabase/table
```

É possível verificar diversas tabelas listando-as na linha de comando ou usando:

```
myisamchk -analyze pathtomysqldatabase/*.MYI
```

É possível verificar todas as tabelas em todos os bancos de dados executando:

```
myisamchk -analyze pathtomysqldatadirectory/*/*.MYI
```

Ou então você pode listar as tabelas em uma instrução ANALYZE TABLE dentro do monitor MySQL:

```
analyze table customers, orders, order_items, books;
```

Terceiro, você pode querer considerar acrescentar um novo índice à tabela. Se essa consulta for lenta e comum, seria importante considerar seriamente. Se for uma consulta de uma única vez que nunca mais será utilizada, como um relatório obscuro solicitado uma única vez, essa técnica não valerá o esforço porque tornará tudo lento. Veremos como utilizá-la na próxima seção.

Acelerando consultas com índices

Se você estiver na situação mencionada anteriormente, em que a coluna possible_keys de um EXPLAIN contém alguns valores NULL, talvez seja capaz de melhorar o desempenho da consulta adicionando um índice à tabela em questão. Se a coluna que está utilizando em sua cláusula WHERE for adequada para indexar, você pode criar um novo índice para ela utilizando ALTER TABLE assim:

```
ALTER TABLE tabela ADD INDEX (coluna );
```

Otimizando o banco de dados

Além das dicas anteriores de otimização de consulta, há poucas coisas que podem ser feitas para aumentar de modo geral o desempenho do banco de dados MySQL.

Otimização de projeto

Basicamente, você deseja que tudo no seu banco de dados seja o menor possível. Você pode alcançar isso em parte com um projeto decente que minimize redundância. Você também pode alcançar isso utilizando o menor tipo de dados possível para colunas. Você também deve minimizar NULLs onde for possível e tornar sua chave primária a menor possível.

Evite colunas de comprimento variável se possível (como VARCHAR, TEXT e BLOB). Se suas tabelas tiverem campos de tamanho fixo elas serão mais rápidas mas ocuparão um pouco mais de espaço.

Permissões

Além de utilizar sugestões mencionadas na seção anterior em EXPLAIN, você pode melhorar a velocidade de consultas simplificando suas permissões. Discutimos anteriormente a maneira como con-

sultas são verificadas com o sistema de permissão antes de serem executadas. Quanto mais simples for esse processo, mais rapidamente sua consulta executará.

Otimização de tabela

Se uma tabela foi utilizada durante um período de tempo, os dados podem se tornar fragmentados à medida que atualizações e exclusões forem processadas. Isso aumentará o tempo que leva para localizar coisas nessa tabela. Você pode corrigir isso utilizando a instrução

```
OPTIMIZE TABLE nomedatabela;
```

ou digitando

```
myisamchk -r table
```

no prompt de comando.

Você também pode utilizar o utilitário myisamchk para classificar um índice de tabela e os dados de acordo com esse índice, assim:

```
myisamchk -sort-index -sort-records=1 caminho_do_diretório_de_dados_do_mysql/*/*.MYI
```

Utilizando índices

Utilize índices onde requerido para acelerar suas consultas. Mantenha-os simples e não crie índices que sejam utilizados por suas consultas. Você pode verificar que índices estão sendo utilizados executando EXPLAIN como mostrado anteriormente.

Utilize valores padrão

Onde for possível, utilize valores padrão para colunas e somente insira dados se eles diferirem do padrão. Isso reduz o tempo necessário para executar a instrução INSERT.

Outras dicas

Há muitos outros ajustes menores para melhorar o desempenho em situações particulares e quando você tiver necessidades particulares. O Web site do MySQL oferece um bom conjunto de dicas adicionais. Você pode localizá-lo em:

http://www.mysql.com.

Fazendo backup do banco de dados MySQL

No MySQL, há duas maneiras de fazer um backup.

A primeira maneira é bloquear as tabelas enquanto você copia os arquivos físicos, utilizando um comando LOCK TABLES. Isso tem a sintaxe:

```
LOCK TABLES tabela tipo_de_bloqueio [, tabela tipo_de_bloqueio ...]
```

Cada *tabela* deve ser o nome de uma tabela e o *tipo_de_bloqueio* deve ser READ ou WRITE. Para um backup, você só deve precisar de um bloqueio READ. É necessário executar um comando FLUSH TABLES; para certificar-se de que quaisquer mudanças nos índices foram gravadas em dico antes de fazer o backup.

Usuários e scripts ainda podem executar consultas apenas para leitura enquanto você faz o backup. Se você tiver um volume razoável de consultas que alterem o banco de dados, como pedidos de clientes, essa solução não é prática.

O segundo método, e superior, é utilizar o comando mysql_dump. O uso típico é algo como:

```
mysqldump --opt --all-databases > all.sql
```

Isso fará um dump de (copiará) um conjunto de toda a SQL exigida para reconstruir o banco de dados para o arquivo chamado `all.sql`.

Você então deve parar o processo do `mysqld` por um momento e reiniciá-lo com a opção `--log-bin[=`*`arquivodelog`*`]`. As atualizações armazenadas no arquivo de log lhe fornecerão as alterações feitas desde seu dump. (Obviamente você deve fazer o backup dos arquivos de log em qualquer backup normal de arquivos.)

Um terceiro método é utilizar o script `mysqlhotcopy`. Você pode chamá-lo com:

```
mysqlhotcopy database /path/for/backup
```

Então, siga o processo de iniciar e parar o banco de dados conforme descrito anteriormente.

Um método final de backup (e failover) é manter uma cópia réplica do banco de dados. A replicação será discutida posteriormente neste capítulo.

Restaurando o banco de dados MySQL

Se você precisar restaurar seu banco de dados MySQL, novamente há algumas abordagens. Se o problema for uma tabela corrompida, você poderá executar `myisamchk` com a opção `-r` (repair).

Se tiver utilizado o primeiro método para backup, você poderá copiar os arquivos de dados de volta para os mesmos locais em uma nova instalação de MySQL.

Se tiver utilizado o segundo método para backup, há alguns passos. Primeiro, você precisa executar as consultas em seu arquivo de dump. Isso reconstruirá o banco de dados até o ponto em que fez o dump desse arquivo. Segundo, você precisará atualizar o banco de dados até o ponto armazenado nos arquivos de log.

Você pode fazer isso executando o comando:

```
mysqlbinlog nome_do_host-bin.[0-9]* | mysql
```

As informações adicionais sobre o processo de backup e recuperação de MySQL podem ser encontradas no Web site do MySQL:

http://www.mysql.org

Implementando a replicação

A replicação é uma tecnologia que permite ter diversos servidores de banco de dados oferecendo os mesmos dados. Dessa forma, você pode carregar o compartilhamento e melhorar a confiança do sistema; se um servidor sair do ar, os outros ainda podem ser consultados. Uma vez configurado, também pode ser usado para fazer backups.

A idéia básica é ter um servidor mestre e acrescentá-lo a um número de escravos. Cada um dos escravos espelha o mestre. Ao configurar os escravos inicialmente, você faz uma cópia instantânea de todos os dados no mestre. Depois disso, os escravos fazem atualizações a partir do mestre. Este transmite detalhes das consultas que foram executadas a partir do log binário e os escravos as reaplicam aos dados.

A maneira comum de usar essa configuração é aplicar consultas de gravação ao mestre e consultas de leitura aos escravos, o que é reforçado pela lógica de sua aplicação. Arquiteturas mais complexas são possíveis, como ter vários mestres, mas consideraremos apenas a configuração para o exemplo mais comum.

É importante observar que os escravos geralmente não possuem dados que sejam tão atualizados quanto os do mestre. Isso ocorre em qualquer banco de dados distribuído.

Para começar a configurar uma arquitura mestre e escravo, é preciso certificar-se de que o log binário esteja ativado no mestre; esse assunto é discutido no Apêndice A.

É necessário editar o arquivo my.ini ou my.cnf nos servidores mestre e escravo. No mestre, faça as seguintes configurações:

```
[mysqld]
log-bin
server-id=1
```

A primeira configuração ativa o log binário (então, você já deve ter essa; se não, acrescente-a agora). A segunda configuração fornece ao servidor mestre um ID único. Cada um dos escravaos também precisa de um ID; portanto, acrescente uma linha semelhante aos arquivos my.ini/my.cnf em cada um dos escravos. Os números devem ser únicos e diferentes! Por exemplo, seu primeiro escravo pode ter server-id=2; o seguinte, server-id=3 etc.

Configurando o mestre

No mestre, você precisa criar um usuário para os escravos se conectarem como. Existe um nível especial de privilégio para escravos chamado *escravo de replicação*. Dependendo de como você planeja fazer a transferência inicial dos dados, é necessário conceder alguns privilégios adicionais.

Na maioria dos casos, você usará um instantâneo de banco de dados para transferir os dados, e, nesse caso, é necessário apenas o privilégio escravo de replicação. Se você decidir usar o comando LOAD DATA FROM MASTER para transferir dados (veremos esse assunto na próxima seção), esse usuário também precisará de privilégios RELOAD, SUPER e SELECT, mas apenas para a configuração inicial. De acordo com o princípio do menor privilégio, discutido no Capítulo 9, é importante anular esses outros privilégios depois que o sistema estiver executando com sucesso.

Crie um usuário no mestre. Você pode nomeá-lo como quiser e fornecer a senha desejada, mas deve anotar o nome de usuário e senha escolhidos. Em nosso exemplo, chamamos o usuário de rep_slave:

```
grant replication slave
on *.*
to 'rep_slave'@'%' identified by 'senha';
```

Obviamente, você deve mudar a senha para algo diferente.

Realizando a transferência inicial de dados

Você pode transferir os dados do mestre para o escravo de várias maneiras. A mais simples é configurar os escravos (descrito na próxima seção) e então executar uma instrução LOAD DATA FROM MASTER. O problema com essa abordagem é que ela bloqueará as tabelas no mestre enquanto os dados estão sendo transferidos, o que pode levar algum tempo; portanto, não recomendamos. (Você pode escolher essa opção apenas se estiver usando tabelas MyISAM.)

Generalmente, é melhor obter um instantâneo do banco de dados no momento atual. Você pode fazer isso usando os procedimentos descritos neste capítulo para fazer backups. Primeiro é preciso nivelar as tabelas na seguinte instrução:

```
flush tables with read lock;
```

A razão para o bloqueio de leitura é que você precisa registrar o local em que o servidor está em seu log binário, quando o instantâneo foi tirado. Para isso, execute a seguinte instrução:

```
show master status;
```

A saída deverá ser parecida com isto:

```
+----------------------+----------+--------------+------------------+
| File                 | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------------------+----------+--------------+------------------+
| laura-ltc-bin.000001 |       95 |              |                  |
+----------------------+----------+--------------+------------------+
```

Observe File e Position (arquivo e posição); você precisará dessa informação para configurar os escravos.

Agora tire o seu instantâneo e desbloqueie as tabelas com a seguinte instrução:

```
unlock tables;
```

Se estiver utilizando tabelas InnoDB, a maneira mais fácil é usar a ferramenta InnoDB Hot Backup, disponível no Innobase Oy em http://www.innodb.com. Esse não é um software gratuito, portanto, existe um custo de licença. Você também pode seguir o procedimento descrito aqui e desativar o servidor MySQL e copiar todo o diretório para o banco de dados que deseja reproduzir antes de reiniciar o sevidor e desbloquear as tabelas.

Configurando o(s) escravo(s)

Há duas opções para configurar o(s) escravo(s). Se você tirou um instantâneo de seu banco de dados, comece instalando-o em um servidor mestre.

Em seguida, execute as seguintes consultas em seu escravo:

```
change master to
master-host='servidor',
master-user='usuário',
master-password='senha',
master-log-file='arquivo de log',
master-log-pos=pos de log;
start slave;
```

É preciso preencher os dados indicados em itálico. O *servidor* é o nome do servidor mestre. O *usuário* e *senha* vêm da instrução `GRANT` executada no servidor mestre. *arquivo de log* e *pos do log* vêm da saída da instrução `SHOW MASTER STATUS` executada no servidor mestre.

Agora tudo deve estar funcionando.

Se você não tirou o instantâneo, pode carregar os dados do mestre depois de executar a consulta anterior usando com a seguinte instrução:

```
load data from master;
```

Leitura adicional

Nesses capítulos no MySQL, focalizamos o uso e as partes do sistema mais relevantes para o desenvolvimento de aplicações para a Web e como associar o MySQL ao PHP. Se quiser saber mais, particularmente com referência a aplicações não-Web ou administração do MySQL, você pode visitar o Web site do MySQL em:

http://www.mysql.com

Você também pode querer consultar o livro *MySQL Administrator's Guide* da MySQL Press ou o livro de Paul Dubois *MySQL, Third Edition*, disponível na Developer's Library.

A seguir

No próximo capítulo veremos alguns recursos avançados de MySQL que são úteis ao escrever aplicações Web, como usar os diferentes mecanismos de armazenamento, transações e procedures armazenadas.

13

Programação avançada do MySQL

Neste capítulo, você aprenderá alguns tópicos avançados de MySQL, incluindo tipos de tabela, transações e procedures armazenadas.

Os tópicos-chave deste capítulo incluem:

- A instrução LOAD DATA INFILE;
- Mecanismos de armazenamento;
- Transações;
- Chaves estrangeiras;
- Procedures armazenadas.

A instrução **LOAD DATA INFILE**

Um recurso útil do MySQL que ainda não discutimos é a instrução LOAD DATA INFILE. Você pode usá-la para carregar dados a partir de um arquivo em uma tabela. Ela é executada de forma bastante rápida. Esse comando flexível possui muitas opções, mas o uso típico se parece com o seguinte:

```
LOAD DATA INFILE "newbooks.txt" INTO TABLE books;
```

Essa linha lê dados de linha do arquivo newbooks.txt para a tabela books. Por default, os campos de dados no arquivo devem ser separados por tabulações e devem estar entre aspas simples, e cada linha deve ser separada por um newline (\n). Caracteres especiais devem ser "escapados" com uma barra invertida (\). Todas essas características são configuráveis com as diversas opções da instrução LOAD; consulte o manual MySQL para obter mais detalhes.

Para usar a instrução LOAD DATA INFILE, um usuário deve ter o privilégio FILE, discutido no Capítulo 9.

Mecanismos de armazenamento

O MySQL suporta diversos mecanismos diferentes de armazenamento, às vezes chamados de tipos de tabela. Isso significa que você tem uma opção com relação à implementação de tabelas. Cada tabela em seu banco de dados pode utilizar um diferente mecanismo de armazenamento, e você pode facilmente converter entre eles.

Você pode escolher um tipo ao criar uma tabela utilizando:

```
CREATE TABLE table TYPE=type ....
```

Os tipos de tabela possíveis são:

- **MyISAM** – Esse tipo é o default e o que estamos utilizando até agora no livro. Ele é baseado no tipo ISAM (*Indexed Sequential Access Method*) tradicional, um método padrão para armazenar registros e arquivos. MyISAM adiciona diversas vantagens ao tipo ISAM. Comparado com outros mecanismos de armazenamento, MyISAM possui mais ferramentas para verificar e consertar tabelas. As tabelas MyISAM podem ser comprimidas e suportam busca de texto completo. Elas não são seguras de transações e não suportam chaves estrangeiras.
- **ISAM** – Como foi descrito no parágrafo anterior, o uso de tabelas ISAM está condenado.
- **MEMORY** (conhecido anteriormente como **HEAP**) – Tabelas dessa tipo são armazenadas na memória, e seus índices estão em hash. Isso as torna extremamente mais rápidas, mas, quando ocorre um crash, os dados são perdidos. Essas características tornam as tabelas MEMORY ideais para armazenar dados temporários ou derivados. Você deve especificar a instrução `MAX_ROWS in the CREATE TABLE`; caso contrário, essas tabelas podem consumir toda a memória. Além disso, elas não podem ter colunas `BLOB`, `TEXT` ou `AUTO INCREMENT`.
- **MERGE** – Essas tabelas permitem tratar um conjunto de tabelas MyISAM como único, com o propósito de consulta. Dessa forma, você pode tratar limites máximos de tamanho de arquivo em alguns sistemas operacionais.
- **BDB** – Essas tabelas são seguras de transação; ou seja, fornecem capacidade `COMMIT` e `_ROLLBACK`. Elas são mais lentas do que as MyISAM, mas obviamente oferecem todas as vantagens de se usar transações. Elas são baseadas em Berkeley DB.
- **InnoDB** – Essas tabelas também são seguras de transação, e tem as mesmas características que as BDB. Elas também suportam chaves estrangeiras. As tabelas InnoDB são mais rápidas e possuem mais recursos do que as BDB; portanto, se você precisar de um mecanismo de armazenamento de transação segura, esse é o que recomendamos.

Na maioria das aplicações Web, geralmente você usa tabalas MyISAM ou InnoDB ou um misto das duas.

Você deveria preferir MyISAM ao utilizar muitos `SELECT`s ou `INSERT`s em uma tabela (mas não os dois juntos) porque é a mais rápida para isso. Para muitas aplicações Web, como catálogos, MyISAM é a melhor escolha. Você também deve usar MyISAM se precisar de capacidades de busca de texto completo. InnoDB é ideal quando as transações são importantes, como tabelas que armazenam dados financeiros ou em situações em que `INSERT`s e `SELECT`s estão sendo usados, como fóruns ou sistemas de mensagens on-line.

Você pode utilizar tabelas MEMORY para usar tabelas temporárias ou implementar visualizações, e as tabelas MERGE, se precisar lidar com tabelas MyISAM muito grandes.

É possível mudar o tipo de tabela depois de sua criação com a instrução `ALTER TABLE`, da seguinte maneira:

```
alter table orders type=innodb;
alter table order_items type=innodb;
```

Utilizamos tabelas MyISAM na maior parte deste livro. Agora vamos dedicar mais tempo ao uso de transações e as formas como são implementadas em tabelas InnoDB.

Transações

Transações são mecanismos para garantir consistência do banco de dados quando ocorrer um erro ou o servidor der problema. Nas seções a seguir, você aprenderá o que são transações e como implementá-las com InnoDB.

Entendendo as definições de transação

Primeiro, vamos definir o termo. *Transação* é uma consulta ou um conjunto de consultas que deve ser completamente executada no banco de dados ou então não ser executada de forma alguma. Portanto, o banco de dados fica em um estado consistente, tenha ou não a transação sido completada.

Para ver por que essa capacidade deve ser importante, considere um banco de dados de transações bancárias. Imagine a situação em que você deseja transferir dinheiro de uma conta para outra. Essa ação envolve remover o dinheiro de uma conta e colocá-lo em outra, o que exigiria pelo menos duas consultas. É muito importante que essas duas consultas sejam executadas ou então nenhuma seja. Por exemplo, se você tirar dinheiro de uma conta e a luz acaba antes de colocá-lo na outra, o que acontece? O dinheiro simplesmente desaparece?

Você já deve ter ouvido a expressão *ACID compliance*. ACID é uma forma de descrever quatro requisitos que as transações devem cumprir:

- **Atomicidade** – Uma transação deve ser atômica; ou seja, ou ela é completamente executada, ou então não deve ser executada de forma alguma.
- **Consistência** – Uma transação deve deixar o banco de dados em um estado consistente.
- **Isolamento** – Transações incompletas não devem ser visíveis a outros usuários do banco de dados; ou seja, até que se completem, as transações devem se manter isoladas.
- **Durabilidade** – Uma vez que tenha sido gravado no banco de dados, a transação deve ser permanente.

Diz-se que uma transação que tenha sido permanentemente gravada no banco de dados está *concluída*. Quando ela não tiver sido gravada no banco de dados – de modo que este seja redefinido para o estado em que estava antes de a transação ter começado – é referida como *revertida (roll back)*.

Utilizando transações com InnoDB

Por default, o MySQL é executado em um *modo de autocommit*. Isso significa que cada instrução executada é imediatamente gravada no banco de dados (concluída). Se você estiver utilizando um tipo de tabela de transação segura, é muito provável que não queira esse comportamento.

Para desativar o autocommit na sessão atual, digite:

```
set autocommit=0;
```

Se o autocommit estiver ativado, você precisa começar uma transação com a instrução:

```
start transaction;
```

Se estiver desativado, esse comando não é necessário, porque uma transação será iniciada automaticamente para você quando digitar uma instrução SQL.

Depois que terminou de inserir as instruções que formam a transação, você pode enviá-la ao banco de dados apenas digitando:

```
commit;
```

Se mudar de idéia, você pode reverter ao estado anterior do banco de dados digitando:

```
rollback;
```

Até ser finalizada a transação, ela não será visível a outros usuários ou em outras sessões.

Vamos ver um exemplo. Execute as instruções `ALTER TABLE` na seção anterior do capítulo em seu banco de dados `books`, da seguinte forma, se ainda não o fez:

```
alter table orders type=innodb;
alter table order_items type=innodb;
```

Essas instruções convertem duas tabelas para tabelas InnoDB. (Você pode convertê-las de volta se quiser executando a mesma instrução com type=MyISAM.)

Agora abra duas conexões ao banco de dados books. Em uma, acrescente um novo registro de pedido ao banco de dados:

```
insert into orders values
(5, 2, 69.98, '2004-06-18');
insert into order_items values
(5, '0-672-31697-8', 1);
```

Agora verifique se você pode ver o novo pedido:

```
select * from orders where orderid=5;
```

Você deve ver o pedido exibido:

```
+---------+------------+--------+------------+
| orderid | customerid | amount | date       |
+---------+------------+--------+------------+
|       5 |          2 |  69.98 | 2004-06-18 |
+---------+------------+--------+------------+
```

Deixando a conexão aberta, vá à outra conexão e execute a mesma consulta select. O pedido não deve estar visível:

```
Empty set (0.00 sec)
```

(Se puder vê-lo, provavelmente esqueceu de desativar o autocommit. Verifique isso e se converteu a tabela para o formato InnoDB.) O problema é que a transação ainda não foi concluída. (Esse é um bom exemplo de isolamento de transação.)

Agora volte à primeira conexão e conclua a transação:

```
commit;
```

Agora você deve conseguir recuperar a linha na outra conexão.

Chaves estrangeiras

O InnoDB também oferece suporte a chaves estrangeiras. Você deve se lembrar de que discutimos o conceito de chaves estrangeiras no Capítulo 8. Ao usar tabelas MyISAM, não há como forçar chaves estrangeiras.

Considere, por exemplo, inserir uma linha na tabela order_items. É preciso incluir um orderid válido. Utilizando MyISAM, você precisa garantir a validade do orderid que inseriu em algum lugar na aplicação. Usando chaves estrangeiras no InnoDB, você pode deixar que o banco de dados verifique isso para você.

Como conseguir isso? Para criar a tabela inicialmente utilizando uma chave estrangeira, você pode mudar a instrução DDL da tabela assim:

```
create table order_items
( orderid int unsigned not null references orders(orderid),
  isbn char(13) not null,
  quantity tinyint unsigned,
  primary key (orderid, isbn)
) type=InnoDB;
```

Adicionamos as palavras references orders(orderid) depois de orderid. Isso significa que essa coluna é uma chave estrangeira que deve conter um valor da coluna orderid na tabela orders

Finalmente, adicionamos o tipo de tabela type=InnoDB no final da declaração. Isso é necessário para que as chaves estrangeiras funcionem.

Você também pode fazer essas alterações à tabela existente utilizando as instruções ALTER TABLE, da seguinte forma:

```
alter table order_items type=InnoDB;
alter table order_items
add foreign key (orderid) references orders(orderid);
```

Para ver se essa mudança funcionou, tente inserir uma linha com um orderid para a qual não haja linha correspondente na tabela orders:

```
insert into order_items values
(77, '0-672-31697-8', 7);
```

Você deve receber um erro semelhante a

```
ERROR 1216 (23000): Cannot add or update a child row:
a foreign key constraint fails
```

Procedures armazenadas

O MySQL5, em alfa enquanto escrevíamos este livro, apresenta as procedures armazenadas. Essa deve ser a versão em produção em 2005. Nas seções seguintes, veremos o recurso de procedures armazenadas, que será a principal novidade na versão 5.0.

Um procedure armazenada é uma função programática que é criada e armazenada dentro do MySQL. Pode consitir em instruções SQL e diversas estruturas de controle especiais. Pode ser útil quando você quiser realizar a mesma função de diferentes aplicações ou plataformas, ou como uma forma de encapsular funcionalidade. As procedures armazenadas em um banco de dados podem ser vistas como orientação a objetos em programação. Elas permitem controlar a forma como os dados são acessados.

Vamos começar examinando um exemplo simples.

Exemplo básico

A Listagem 13.1 mostra a declaração de uma procedure armazenada.

Listagem 13.1 **basic_stored_procedure.sql – Declarando uma procedure armazenada**

```
# Basic stored procedure example
delimiter //

create procedure total_orders (out total float)
BEGIN
select sum(amount) into total from orders;
END
//

delimiter;
```

Vamos percorrer o código linha por linha.

A primeira instrução

```
delimiter //
```

muda o delimitador da instrução de seu valor atual – normalmente um ponto-e-vírgula, a menos que você tenha mudado anteriormente – para uma barra dupla. Isso é feito para que você possa usar o delitador ponto-e-vírgula dentro da procedure armazenada à medida que digita o código para ela sem que o MySQL tente executar o código imediatamente.

A linha

```
create procedure total_orders (out total float)
```

cria a procedure real. O nome dessa procedure é `total_orders`. Ela possui um parâmetro único chamado `total`, que é o valor que você vai calcular. A palavra `OUT` indica que esse parâmetro está sendo passado ou retornado.

Os parâmetros também podem ser declarados como `IN`, o que significa que um valor está sendo passado à procedure, ou `INOUT`, quando um valor está passado mas pode ser alterado pela procedure.

A palavra `float` indica o tipo do parâmetro. Nesse caso, você retorna um total de todos os pedidos na tabela `orders`. O tipo da coluna `orders` é `float`, então o tipo retornado também é `float`. Os tipos de dados aceitáveis são mapeados para os tipos de coluna disponíveis.

Se quiser mais de um parâmetro, você pode fornecer uma lista de parâmetros separada por vírgulas, como faria no PHP.

O conteúdo da procedure fica entre as instruções `BEGIN` e `END`. Elas são semelhantes às chaves no PHP ({ }) porque delimitam um bloqueio de instrução.

No conteúdo, você simplesmente executa uma instrução `SELECT`. A única diferença do normal é que você inclui a cláusula `into total` para carregar o resultado da consulta para o parâmetro `total`.

Depois de ter declarado a procedure, faça com que o delimitador volte a ser um ponto-e-vírgula, com a linha:

```
delimiter;
```

Depois que a procedure tiver sido declarada, você pode chamá-la usando a palavra-chave `call`, da seguinte forma:

```
call total_orders(@t);
```

Essa instrução chama os pedidos totais e passa uma variável para armazenar o resultado. Para ver o resultado, é preciso então examinar a variável:

```
select @t;
```

O resultado deve ser parecido com:

```
+-----------------+
| @t              |
+-----------------+
| 289.92001152039 |
+-----------------+
```

Você pode criar uma função de uma maneira semelhante a criar uma procedure. Uma função aceita parâmetros de entrada (apenas) e retorna um único valor. (Enquanto escrevíamos este livro, funções não podiam referenciar tabelas, mas essa limitação deve mudar antes de a versão 5.0 tornar-se a versão a ser produzida.)

A sintaxe básica para essa tarefa é quase igual. Uma função simples é mostrada na Listagem 13.2.

Listagem 13.2 **basic_function.sql** – Declarando uma função armazenada

```
# Basic syntax to create a function

delimiter //

create function add_tax (price float) returns float
return price*1.1;
//

delimiter;
```

Como você pode ver, esse exemplo utiliza a palavra-chave `function` em vez de `procedure`. Existem algumas outras diferenças.

Os parâmetros não precisam ser especificados como `IN` ou `OUT` porque todos são `IN`, ou seja, parâmetros de entrada. Depois da lista de parâmetros, você pode ver a cláusula `returns float`. Ela especifica o tipo de valor de retorno. Novamente, esse valor pode ser qualquer um dos tipos válidos MySQL.

Você retorna um valor utilizando a instrução `return`, como faria no PHP.

Observe que esse exemplo não utiliza as instruções `BEGIN` e `END`. Você poderia utilizá-las, mas não é necessário. Assim como no PHP, se um bloco de instrução contém apenas uma instrução, você não precisa marcar o início e o final dela.

Chamar uma função é um pouco diferente de chamar uma procedure. Você pode chamar uma função armazenada da mesma forma que uma função embutida. Por exemplo:

```
select add_tax(100);
```

Essa instrução retornaria a seguinte saída:

```
+-------------+
| add_tax(100) |
+-------------+
|         110 |
+-------------+
```

Depois de ter definido procedures e funções, você pode visualizar o código utilizado para defini-las usando, por exemplo,

```
show create procedure total_orders;
```

ou

```
show create function addtax;
```

Pode excluí-las com

```
drop procedure total_orders;
```

ou

```
drop function add_tax;
```

As procedures armazenadas vêm com a capacidade de controlar estruturas, variáveis, handlers `DECLARE` (como exceções) e um conceito importante chamado *cursores*. Veremos cada um deles nas seções a seguir.

Variáveis locais

É possível declarar variáveis locais dentro de um bloco `begin...end` utilizando uma instrução `declare`. Por exemplo, você poderia alterar a função `add_tax` para utilizar uma variável local a fim de armazenar a porcentagem da tributação (*tax rate*), como mostra a Listagem 13.3.

Listagem 13.3 **`basic_function.sql` – Declarando uma função armazenada com variáveis**

```
# Basic syntax to create a function

delimiter //

create function add_tax (price float) returns float
begin
  declare tax float default 0.10;
```

Listagem 13.3 **Continuação**

```
  return price*(1+tax);
end
//
delimiter;
```

Como você pode ver, declaramos a variável utilizando `declare`, seguido pelo nome da variável, seguido pelo tipo. A cláusula default é opcional e especifica um valor inicial para a variável. Então você pode utilizar a variável conforme esperado.

Cursores e estruturas de controle

Vamos considerar um exemplo mais complexo. Para isso, você escreverá uma procedure armazenada que calcula que pedido teve a maior quantidade e retorna o `orderid`. (Obviamente, você poderia calcular essa quantidade facilmente com uma única consulta, mas esse simples exemplo ilustra como utilizar cursores e estruturas de controle.) O código para essa procedure armazenada é mostrado na Listagem 13.4.

Listagem 13.4 **`control_structures_cursors.sql` – Utilizando cursores e loops para processar um conjunto de resultados**

```
# Procedure to find the orderid with the largest amount
# could be done with max, but just to illustrate stored procedure principles

delimiter //

create procedure largest_order(out largest_id int)
begin
  declare this_id int;
  declare this_amount float;
  declare l_amount float default 0.0;
  declare l_id int;

  declare done int default 0;
  declare continue handler for sqlstate '02000' set done = 1;
  declare c1 cursor for select orderid, amount from orders;

  open c1;
  repeat
    fetch c1 into this_id, this_amount;
    if not done then
      if this_amount > l_amount then
        set l_amount=this_amount;
        set l_id=this_id;
      end if;
    end if;
   until done end repeat;
  close c1;

  set largest_id=l_id;

end
//

delimiter;
```

Esse código utiliza as estruturas de controle (condicionais e de loop), cursores e declara handlers. Vamos analisá-lo linha por linha.

No início da procedure, você declara algumas variáveis locais para utilizar dentro da procedure. As variáveis `this_id` e `this_amount` armazenam os valores de `orderid` e `amount` na linha atual. As variáveis `l_amount` e `l_id` são para armazenar a maior quantidade de pedidos e o ID correspondente. Como você calculará a maior quantidade comparando cada valor com o maior valor atual, inicialize essa variável em zero.

A próxima variável declarada é `done`, inicializada em zero (`false`). Ela é o seu flag de loop. Quando terminarem as linhas, defina essa variável para 1 (`true`):

A linha

```
declare continue handler for sqlstate '02000' set done = 1;
```

é chamada de *handler de declaração*. É semelhante a uma exceção em procedures armazenadas. Enquanto escrevíamos este livro, você podia implementar handlers de continuação e handlers de saída. Os handlers de continuação, como o mostrado aqui, tomam a ação especificada e então continuam a execução da procedure. Os handlers de saída saem do bloco `begin...end` mais próximo.

A próxima parte do handler de declaração especifica quando o handler será chamado. Nesse caso, será chamado quando `sqlstate '02000'` for alcançado. Você pode se perguntar o que isso significa porque parece muito enigmático! Significa que ele será chamado quando nenhuma linha for encontrada. Você processa um conjunto de resultados linha por linha e quando terminarem as linhas a serem processadas, esse handler será chamado. Também é possível especificar `FOR NOT FOUND` da mesma forma. Outras opções são `SQLWARNING` e `SQLEXCEPTION`.

O próximo elemento é o *cursor*. Ele não é diferente de um array; recupera um conjunto de resultados para uma consulta (conforme retornado por `mysqli_query( )`) e permite processar uma linha por vez (como, por exemplo, com `mysqli_fetch_row( )`). Considere este cursor:

```
declare c1 cursor for select orderid, amount from orders;
```

Ele é chamado `c1`. É apenas uma definição do que manterá. A consulta ainda não foi executada.

A linha

```
open c1;
```

na verdade executa a consulta. Para obter cada linha dos dados, é preciso executar uma instrução `fetch`. Faça isso no loop `repeat`. Nesse caso, o loop se parece com:

```
repeat
...
until done end repeat;
```

Observe que a condição (`until done`) não é verificada até o final. As procedures armazenadas também suportam loops `while`, do tipo:

```
while condição do
...
end while;
```

Existem também loops `loop`, do tipo:

```
loop
...
end loop
```

Eles não possuem condições embutidas, mas podem gerar saída com uma instrução `leave;`.

Observe que não há loops `for`.

Continuando com o exemplo, a próxima linha de código revoga uma linha de dados:

```
fetch c1 into this_id, this_amount;
```

Essa linha recupera uma linha da consulta do cursor. Os dois atributos recuperados pela consulta são armazenados nas duas variáveis locais especificadas.

Verifique se a linha foi recuperada e então compare a quantidade de loop atual com a maior quantidade armazenada, utilizando as duas instruções `IF`:

```
if not done then
 if this_amount > l_amount then
   set l_amount=this_amount;
   set l_id=this_id;
 end if;
end if;
```

Observe que os valores das variáveis são definidos pela instrução set.

Além de `if...then`, as procedures armazenadas também suportam uma construção `if...then...else` com o seguinte formato:

```
if condition then
   ...
   [elseif condition then]
   ...
   [else]
   ...
end if
```

Também existe uma instrução case, que possui o seguinte formato:

```
case value
   when value then statement
   [when value then statement ...]
   [else statement]
end case
```

De volta ao exemplo, depois que o loop termina, é preciso fazer um pouco de limpeza:

```
close c1;
set largest_id=l_id;
```

A instrução `close` fecha o cursor.

Finalmente, defina o parâmetro `OUT` para o valor que você calculou. Você não pode utilizar o parâmetro como uma variável temporária, apenas para armazenar o valor final. (Esse uso é semelhante a algumas outras linguagens de programação, como Ada.)

Se criar essa procedure conforme descrita aqui, você pode chamá-la como fez com a outra procedure:

```
call largest_order(@l);
select @l;
```

A saída deve ser semelhante a:

```
+------+
| @l   |
+------+
| 3    |
+------+
```

Você mesmo pode verificar se o cálculo está correto.

Leitura adicional

Neste capítulo, fizemos um passeio pela funcionalidade das procedures armazenadas. Você pode descobrir mais sobre elas no manual do MySQL. Embora, até o momento em que escrevíamos este

livro, não houvesse muito, sem dúvida a documentação será aprimorada antes que esta versão do MySQL se tornar a versão de produção.

Para obter mais informações sobre `LOAD DATA INFILE`, os diferentes mecanismos de armazenamento e procedures armazenadas, consulte o manual do MySQL.

Se quiser descobrir mais sobre transações e consistência de banco de dados, recomendamos um bom texto básico sobre bancos de dados relacionais, como *An Introduction to Database Systems,* escrito por C.J. Date.

A seguir

Agora que abrangemos os fundamentos do PHP e MySQL, o Capítulo 14 trata de comércio eletrônico e aspectos de segurança para configurar Web sites baseados em bancos de dados.

III

Comércio eletrônico e segurança

14

Mantendo um site de comércio eletrônico

ESTE CAPÍTULO INTRODUZ ALGUMAS DAS QUESTÕES envolvidas em especificar, projetar, construir e manter um site de comércio eletrônico de maneira eficiente. Examinaremos seu plano, possíveis riscos e algumas maneiras de fazer um Web site se pagar.

Os tópicos-chave deste capítulo incluem:

- O que você deseja alcançar com seu site de comércio eletrônico;
- Tipos de Web site comercial;
- Riscos e ameaças;
- Decidindo uma estratégia.

O que você deseja alcançar?

Antes de perder tempo preocupando-se com os detalhes de implementação do Web site, você deve ter firmes objetivos em mente e um plano razoavelmente detalhado que conduza para o atendimento desses objetivos.

Neste livro, fazemos a suposição de que você está construindo um Web site comercial. Então, presumivelmente, ganhar dinheiro é um de seus objetivos.

Há muitas maneiras de adotar uma abordagem comercial para a Internet. Você poderia querer anunciar seus serviços off-line ou vender um produto do mundo real on-line. Talvez você tenha um produto que possa ser vendido e fornecido on-line. Talvez seu site não tenha a intenção de gerar renda diretamente, mas, em vez disso, suportar atividades off-line ou atuar como uma alternativa mais barata de apresentar atividades.

Tipos de Web sites comerciais

Web sites comerciais geralmente realizam uma ou mais das seguintes atividades:

- Publicar informações da empresa por meio de brochuras on-line;
- Fazer pedidos de mercadoria ou serviços;
- Fornecer serviços ou mercadoria digitais;
- Agregar valor a mercadorias ou serviços;
- Cortar custos.

Seções de muitos Web sites se encaixarão em mais de uma dessas categorias. O que segue é uma descrição de cada categoria e a maneira comum de fazer cada uma gerar renda ou outros benefícios para sua organização.

O objetivo desta seção do livro é ajudar a formular seus objetivos. Por que você quer um Web site? Como é construído cada recurso no Web site que vai contribuir para seu negócio?

Publicando informações em brochuras on-line

Quase todos os Web sites comerciais no início da década de 1990 eram simplesmente uma brochura on-line ou uma ferramenta de vendas. Esse tipo de site é ainda a forma mais comum de Web site comercial. Quer por ser uma abordagem inicial para a Web, quer por ser um exercício de baixo custo de publicidade, esse tipo de site faz sentido para muitos negócios.

Um site *brochureware* pode ser desde um cartão de apresentação representado como uma página Web até uma extensa coleção de informações de marketing. Em qualquer caso, o propósito do site, e sua razão financeira de existir, é atrair clientes para entrar em contato com seu negócio. Esse tipo de site não gera nenhuma renda diretamente, mas pode aumentar a renda que seu negócio gera pelos meios tradicionais.

Desenvolver um site assim apresenta poucos desafios técnicos. As questões encontradas são semelhantes àquelas em outros exercícios de marketing. Algumas das armadilhas mais comuns com esse tipo de site incluem:

- Não conseguir fornecer informações importantes;
- Ter uma apresentação pobre;
- Não responder ao feedback gerado pelo site;
- Deixar um site envelhecer;
- Não monitorar o sucesso do site.

Não conseguir fornecer informações importantes

Quais são os prováveis visitantes a buscar quando eles visitarem seu site? Dependendo do quanto eles já conhecem, talvez queiram especificações detalhadas de produto ou talvez só queiram informações muito básicas como detalhes para entrar em contato.

Muitos Web sites não fornecem nenhuma informação útil, ou não possuem informações cruciais. No mínimo, seu site precisa informar os visitantes sobre o que você faz, a que áreas geográficas seus negócios atendem e como entrar em contato.

Ter uma apresentação pobre

"Na Internet, ninguém sabe se você é um cachorro" ou algo que o valha, como diz o velho ditado.[1] Da mesma maneira como empresas de pequeno porte, ou cachorros, podem causar maior e melhor impressão quando estão utilizando a Internet, grandes negócios podem parecer pequenos, não profissionais e não causar impressão com um Web site pobre.

Independente do tamanho da empresa, certifique-se de que seu Web site é de um alto padrão. O texto deve ser escrito e revisado por alguém com um entendimento muito bom do idioma sendo utilizado. As imagens gráficas devem ser claras, limpas e de download rápido. Em um site de negócios, você deve considerar cuidadosamente a utilização de imagens gráficas e cor, e certificar-se de que elas se ajustam à imagem que você quer apresentar. Utilize animação e som criteriosamente, se é que você deve utilizar.

---

[1] É claro que um "velho ditado" sobre a Internet não pode ser realmente muito velho. Isso foi retirado de um desenho de Peter Steiner publicado em 5 de julho de 1993 no *The New Yorker.*

Embora você não seja capaz de fazer seu site parecer o mesmo em todas as máquinas, sistemas operacionais e navegadores, certifique-se de que ele seja visualizável e não forneça erros à maioria dos usuários.

Não responder ao feedback gerado pelo Web site

No bom serviço de atendimento ao cliente é tão vital atrair e conservar clientes na Web como é no mundo do comércio externo. Empresas grandes e pequenas são culpadas de colocar um endereço de e-mail em uma página Web e então esquecer de verificar ou responder a esse correio prontamente.

As pessoas têm diferentes expectativas sobre o tempo de resposta por e-mail daquele que elas têm em relação ao correio convencional. Se não verificar e responder ao correio diariamente, as pessoas acreditarão que a consulta deles não é importante para você.

Os endereços de e-mail nas páginas Web normalmente devem ser genéricos, endereçados a um cargo ou departamento, em vez de a uma pessoa específica. O que acontecerá ao correio enviado para `fred.smith@example.com` quando Fred sair da empresa? O correio para `sales@example.com` provavelmente será passado para seu sucessor. Ele também poderia ser entregue a um grupo de pessoas, o que talvez ajude a assegurar que ele seja respondido prontamente.

Deixar um site envelhecer

Você precisa ter o cuidado de manter seu Web site atualizado. O conteúdo precisa ser alterado periodicamente. As alterações na organização precisam ser refletidas no site. Um site "teia de aranha" desencoraja as pessoas de voltar e as leva a suspeitar de que grande parte das informações agora talvez seja incorreta.

Uma maneira de evitar um site ultrapassado ou envelhecido é atualizar páginas manualmente. Outra é utilizar uma linguagem de criação de scripts como PHP para criar páginas dinâmicas. Se seus scripts têm acesso a informações atualizadas, eles podem gerar constantemente páginas atualizadas.

Não monitorar o sucesso do site

Criar um Web site é algo bom, positivo etc., mas como justificar o esforço e as despesas? Se o site for particularmente para uma empresa grande, surgirá um momento em que você será solicitado a demonstrar ou quantificar seu valor à organização.

Para campanhas tradicionais de marketing, as organizações gastam várias dezenas de milhares de dólares em pesquisa de mercado, tanto antes de lançarem uma campanha como depois para medir sua eficácia. Dependendo da escala e do orçamento de seu empreendimento na Web, essas medidas talvez sejam igualmente apropriadas para ajudar no projeto e na medição de seu site.

As opções mais simples ou mais baratas incluem:

- **Examinar logs de servidor:** Os servidores Web armazenam uma grande quantidade de dados sobre cada solicitação ao seu servidor. Grande parte desses dados é inútil e seu puro volume o torna inútil na sua forma bruta. Para destilar os arquivos de log em um resumo significativo, você precisa de um analisador de arquivo de log. Dois dos programas gratuitos mais famosos são o Analog, que está disponível em http://www.statslab.cam.ac.uk/~sret1/analog e o Webalizer, disponível em http://www.mrunix.net/webalizer/. Programas comerciais como Summary, disponível em http://summary.net, talvez sejam mais abrangentes. Um analisador de arquivo de log mostrará como o tráfego para seu site muda ao longo do tempo e que páginas estão sendo visualizadas.
- **Monitorar vendas:** Sua brochura on-line deve gerar vendas. Você deve ser capaz de estimar seu efeito em vendas comparando níveis de vendas antes e depois do carregamento do site. Isso obviamente torna-se difícil se outros tipos de marketing causam flutuações no mesmo período.

- **Solicitar feedback de usuário:** Se perguntar a eles, seus usuários dirão o que pensam do site. Fornecendo um formulário de retorno ou endereço de e-mail, será possível reunir algumas opiniões úteis. Para aumentar a quantidade de feedback, talvez você queira oferecer uma pequena indulgência, como um número para concorrer a um prêmio atraente para todos os que responderem.
- **Pesquisar usuários representativos:** Manter grupos de foco pode ser uma técnica eficiente para avaliar o site ou mesmo um protótipo de seu futuro site. Para conduzir um grupo de foco, você simplesmente precisa reunir alguns voluntários, encorajá-los a avaliar o site e, então, entrevistá-los para eles avaliarem o site e emitirem suas opiniões.

Os grupos de foco podem ser negócios caros, conduzidos por facilitadores profissionais, que avaliam e verificam participantes potenciais para tentar assegurar que eles representam exatamente a distribuição demográfica e de personalidades na comunidade mais ampla e, então, habilmente entrevistam participantes. Os grupos de foco também podem não custar nada, ser criados e mantidos por um amador e ser preenchidos por uma amostra de pessoas cuja relevância para o mercado-alvo seja desconhecida.

Pagar uma empresa de pesquisa de mercado especializada é uma maneira de obter um bom trabalho de grupo de foco e resultados úteis, mas não é a única maneira. Se você estiver criando seus próprios grupos de foco, escolha um moderador habilidoso. O moderador deve ter excelentes habilidades pessoais e não ter uma opinião ou aposta no resultado da pesquisa. Limite o tamanho do grupo a algo entre seis e dez pessoas. O moderador deve ser auxiliado por um gravador ou secretário a fim de ficar livre para facilitar a discussão. O resultado que você obtém dos grupos será exatamente tão relevante quanto a amostra de pessoas que utiliza. Se você avaliar seu produto somente com amigos e a família do seu pessoal, provavelmente eles não representarão a comunidade geral.

Recebendo pedidos de mercadoria ou serviços

Se seus anúncios publicitários on-line forem atraentes, o próximo passo lógico é permitir que seus clientes façam pedidos enquanto ainda on-line. A equipe de vendas tradicional sabe que é importante fazer o cliente tomar uma decisão agora. Quanto mais tempo você dá às pessoas para reconsiderarem uma decisão de compra, mais chances elas têm de escolher comprar em outro lugar ou mudar de idéia. Se um cliente quer seu produto, seu maior interesse é tornar a compra o mais rápida e fácil possível. Forçar as pessoas a desligarem o modem e ligarem para um número de telefone ou visitarem uma loja coloca obstáculos no caminho delas. Se você tiver anúncios publicitários on-line que convenceram um usuário a comprar, permita que ele compre agora, sem deixar seu Web site.

Receber pedidos em um Web site faz sentido para muitos negócios. Todos os negócios querem receber pedidos. Permitir que sejam feitos pedidos on-line pode gerar vendas adicionais ou reduzir a carga de trabalho dos funcionários de vendas. Obviamente haverá custos envolvidos. Construir um site dinâmico, organizar formas de pagamento e fornecer serviço de atendimento ao cliente, tudo isso custa dinheiro.

Muito do apelo das vendas on-line é que os custos continuam os mesmos independente de serem feitos 1.000 ou 1.000.000 de pedidos. Para que os custos valham a pena, é necessário ter produtos ou serviços que que serão vendidos em número razoável. Antes de se fixar muito na idéia de comércio eletrônico, tente determinar se seus produtos são adequados para um site de comércio eletrônico.

Os produtos e serviços que são comumente comprados utilizando a Internet incluem livros e revistas, software de computador e equipamento, música, vestuário, passagens aéreas e ingressos para eventos de entretenimento.

Não se desespere só porque seu produto não está em uma dessas categorias. Essas categorias já estão abarrotadas com marcas estabelecidas. Entretanto, seria inteligente considerar alguns fatores que tornam esses produtos grandes vendedores on-line.

Idealmente, um produto de comércio eletrônico é não-perecível e facilmente distribuído, caro o bastante para que os custos de entrega pareçam razoáveis, ainda não tão caro que o comprador sinta-se compelido a examinar fisicamente o item antes de comprar.

Os melhores produtos de comércio eletrônico são *commodities*. Se você comprar um abacate, provavelmente vai querer ver o abacate particular e talvez experimentá-lo. Os abacates não são todos idênticos. Uma cópia de um livro, de um CD ou um programa de computador normalmente é idêntica a outras cópias do mesmo título. Os compradores não precisam ver o item particular que comprarão.

Além disso, os produtos de comércio eletrônico devem ser atraentes para as pessoas que utilizam a Internet. Na época da publicação deste livro, essa audiência consiste principalmente em adultos mais jovens empregados e, com renda acima da média, vivendo em áreas metropolitanas. Entretanto, com o passar do tempo, a população on-line está começando a parecer com a população inteira.

Alguns produtos nunca serão refletidos em pesquisas de compras de comércio eletrônico, mas esses produtos ainda são um sucesso. Se você tiver um produto que atraia somente um nicho do mercado, a Internet talvez seja a maneira ideal de alcançar compradores. Mesmo que em sua cidade apenas dez pessoas colecionem gravuras dos anos 80, um site vendendo-as seria bem-sucedido se dez pessoas em cada cidade colecionassem também.

Alguns produtos são improváveis de serem bem-sucedidos como categorias de comércio eletrônico. Itens baratos e perecíveis, como mantimentos, parecem uma escolha pobre, embora isso não tenha impedido empresas de tentar, a maioria sem sucesso. Outras categorias se encaixam muito bem na categoria de sites *brochureware*, mas não são muito propensas a aceitarem pedidos on-line. Itens caros e grandes entram nessa categoria – itens como veículos e bens imobiliários que requerem bastante pesquisa antes de comprar, mas que são muito caros para encomendar sem ver e complicados para entregar.

Há vários obstáculos para convencer um comprador em potencial a completar um pedido. Esses incluem:

- Perguntas não-respondidas;
- Confiança;
- Facilidade de uso;
- Compatibilidade.

Se um usuário fica frustrado com qualquer um desses obstáculos, provavelmente deixará o site sem comprar.

Perguntas não-respondidas

Se um potencial cliente não pode encontrar uma resposta imediata a uma de suas perguntas, esse cliente provavelmente deixará o site. Isso tem várias implicações. Certifique-se de que o site está bem organizado. Uma pessoa que visita o site pela primeira vez pode encontrar o que deseja facilmente? Certifique-se de que o site seja abrangente, sem sobrecarregar os visitantes. Na Web, as pessoas tendem mais a ler rapidamente do que cuidadosamente, então seja conciso. Para a maior parte da mídia publicitária, há limites práticos sobre quanta informação pode ser fornecida. Isso não é verdadeiro para um Web site. Para um Web site, os dois limites principais são o custo de criar e atualizar informações e os limites impostos sobre sua capacidade de organizar, estruturar e conectar as informações de modo que não sobrecarregue os visitantes.

É tentador pensar em um Web site como um vendedor automático, não remunerado, que nunca dorme, mas o serviço de atendimento ao cliente ainda é importante. Encoraje os visitantes a fazerem perguntas. Tente fornecer respostas imediatas ou quase imediatas via telefone, e-mail ou algum outro meio conveniente.

Confiança

Se um visitante não conhece sua marca, por que ele deve confiar em você? Qualquer pessoa pode montar um Web site. As pessoas não precisam confiar em você para ler seu site *brochureware*, mas fazer um pedido exige uma certa confiança. Como um visitante saberá se sua organização é respeitável ou se você é o cachorro mencionado anteriormente?

As pessoas preocupam-se com várias questões ao fazer compras on-line:

- **O que você vai fazer com suas informações pessoais?** Você vai vendê-las a outros, utilizá-las para enviar enormes quantidades de anúncios publicitários ou armazená-las em algum lugar sem proteção de modo que outros possam ganhar acesso a essas informações? É importante informar às pessoas o que você fará e o que não fará com os dados delas. Isso é chamado de *política de privacidade* e deve estar facilmente acessível no site.
- **Você tem um negócio respeitável?** Se seu negócio estiver registrado junto à autoridade relevante em um lugar particular, tiver um endereço físico e um número de telefone e exercer comércio há vários anos, é menos provável que seja um esquema fraudulento do que um negócio que consiste unicamente em um Web site e talvez uma caixa de correio. Certifique-se de exibir esses detalhes.
- **O que acontece se um comprador não estiver satisfeito com uma compra?** Sob quais circunstâncias você fornecerá um reembolso? Quem paga pela entrega? Revendedores pelo sistema de reembolso postal têm tradicionalmente políticas de devolução e reembolso mais liberais do que lojas tradicionais. Muitos oferecem uma garantia de satisfação incondicional. Considere o custo de retornos contra o aumento em vendas que uma política de devolução liberal criará. Seja qual for sua política, certifique-se de que esteja exibida no site.
- **Os clientes deveriam confiar as informações sobre seus cartões de crédito a você?** A única questão importante quanto à confiança para os compradores na Internet é o temor de transmitir os detalhes dos seus cartões de crédito pela Internet. Por essa razão, você precisa tratar os cartões de crédito com segurança e ser visto como tendo perfeito conhecimento de segurança. No mínimo, isso significa utilizar SSL (Secure Sockets Layer) para transmitir os detalhes a partir do navegador do usuário para seu servidor Web e assegurar que este seja competente e seguramente administrado. Discutiremos isso em mais detalhe mais tarde.

Facilidade de uso

Os consumidores variam extensamente em sua experiência com o computador, idioma, conhecimentos gerais, memória e visão. Seu site precisa ser o mais fácil possível de utilizar. O assunto de projeto de interface com o usuário ocuparia muitos livros por si só, mas eis algumas diretrizes:

- **Mantenha seu site o mais simples possível.** Quanto mais opções, anúncios e distrações em cada tela, maiores as chances de um usuário ficar confuso.
- **Mantenha o texto claro.** Utilize fontes claras e não complicadas. Não utilize fontes muito pequenas para texto e lembre-se de que o site terá tamanhos diferentes nos diferentes tipos de máquinas.
- **Torne seu processo de pedidos o mais simples possível.** Tanto a intuição como a evidência disponível suportam a idéia de que quanto mais cliques de mouse os usuários têm de realizar para fazer um pedido, menor a chance de eles completarem o processo. Mantenha o número de passos a um mínimo, mas note que a Amazon.com tem uma patente norte-americana[2] em um processo que utiliza somente um clique, o que eles chamam de 1-Click. Essa patente é fortemente questionada por muitos proprietários de Web site.

---

[2] Número da patente em U.S. Patent and Trademark Office 5.960.411. Método e sistema para fazer uma compra via uma rede de comunicações.

- **Tente não deixar os usuários perdidos.** Forneça marcos e chaves navegacionais para instruir os usuários sobre onde eles estão. Por exemplo, se um usuário estiver dentro de uma subseção do site, destaque a navegação para essa subseção.

Se estiver utilizando um carrinho de compras no qual você fornece um contêiner virtual para clientes acumularem compras antes de finalizarem a venda, mantenha sempre um link para o carrinho de compras visível na tela.

Compatibilidade

Não deixe de testar o site em vários navegadores e sistemas operacionais. Se ele não funcionar para um navegador ou sistema operacional popular, você parecerá não-profissional e perderá uma fatia de seu mercado potencial.

Se o site já estiver operando, os logs do seu servidor Web podem informar quais navegadores os visitantes estão utilizando. De modo geral, se você testar o site nas últimas duas versões do Microsoft Internet Explorer, uma versão recente do Internet Explorer e Safari em um Mac da Apple, a versão atual do Mozilla em Linux e um navegador de texto como Lynx, você será visível para a maioria dos usuários. Tenha o cuidado de visualizar o site em diversas resoluções de tela. É difícil fazer o mesmo site ter a mesma aparência em uma tela que tenha 2.048 pixels e uma que tenha 240.

Tente evitar recursos e instalações que sejam muito recentes, a menos que você esteja disposto a escrever e a manter diversas versões do site.

Fornecendo bens e serviços digitais

Muitos produtos ou serviços podem ser vendidos pela Web e entregues para o cliente via um *courier*. Alguns serviços podem ser imediatamente entregues on-line. Se um serviço ou mercadoria pode ser transmitido por um modem, ele pode ser encomendado, pago e entregue instantaneamente, sem interação humana. O serviço mais óbvio fornecido dessa maneira são informações. Às vezes as informações são inteiramente gratuitas ou patrocinadas por anúncios publicitários. Algumas informações são fornecidas via assinatura ou pagas individualmente.

A mercadoria digital inclui livros eletrônicos (e-books) e música em formatos eletrônicos como MP3. Fotos e desenhos de bancos de imagens (stock library images) podem ser digitalizados e descarregados. O software de computador nem sempre precisa estar em um CD empacotado. Você pode fazer download desse software diretamente. Os serviços que podem ser vendidos dessa maneira incluem acesso à Internet ou hospedagem Web e alguns serviços profissionais que podem ser substituídos por um sistema especializado.

Se você despachar fisicamente um item que foi encomendado no seu Web site, há tanto vantagens como desvantagens em relação aos bens e serviços digitais. Enviar um item físico custa dinheiro. Downloads digitais são quase gratuitos. Isso significa que se você tiver algo que possa ser duplicado e vendido digitalmente, o custo é muito semelhante se você vender um item ou mil itens. Naturalmente, há limites para isso – se alcançar um determinado nível de vendas e tráfego, você precisará investir mais em hardware ou largura de banda.

Os produtos ou serviços digitais podem ser fáceis de vender como compras de impulso. Se uma pessoa faz um pedido de um item físico, em um dia ou mais a mercadoria chega ao comprador. Os downloads são normalmente medidos em segundos ou minutos. Ser imediato pode ser um fardo para comerciantes. Se estiver entregando uma compra digitalmente, você precisa fazer isso na hora. Você não pode administrar manualmente o processo ou distribuir picos de atividade pelo dia. Os sistemas de entrega imediata são, portanto, mais suscetíveis a fraude e mais uma carga sobre os recursos de computação.

Mercadoria e serviços digitais são ideais para comércio eletrônico, mas obviamente só uma faixa limitada de mercadoria e serviços pode ser entregue dessa maneira.

Agregando valor à mercadoria ou aos serviços

Algumas áreas bem-sucedidas de Web sites comerciais realmente não vendem nenhuma mercadoria ou serviços. Serviços como os das empresas de *courier* (sistema de coleta e entrega rápida de encomendas, correspondência etc. como a UPS em www.ups.com ou a Fedex em www.fedex.com) não são geralmente projetados para terem diretamente um lucro. Eles agregam valor aos serviços existentes oferecidos pela organização. Permitir que os clientes monitorem seus pacotes ou saldos bancários pode oferecer uma vantagem competitiva à empresa.

Os fóruns de suporte também entram nessa categoria. Há razões comerciais sadias para fornecer aos clientes uma área de discussão para compartilhar dicas de solução de problemas sobre os produtos da sua empresa. Talvez os clientes sejam capazes de resolver seus problemas examinando soluções oferecidas a outros clientes, clientes internacionais podem obter suporte sem pagar chamadas telefônicas de longa de distância e talvez os clientes sejam capazes de responder a perguntas entre si fora do horário comercial. Fornecer suporte dessa maneira pode aumentar a satisfação dos seus clientes a um baixo custo.

Cortando custos

Uma utilização popular da Internet é cortar custos. Economia poderia resultar do fato de se distribuir informações on-line, facilitar comunicação, substituir serviços ou centralizar operações.

Se atualmente você fornece informações para várias pessoas, possivelmente poderia fazer o mesmo de modo mais econômico via um Web site. Se você estiver fornecendo listas de preço, um catálogo, procedimentos documentados, especificações ou qualquer outro tipo de informação, poderia ser mais barato disponibilizar as mesmas informações pela Web em vez de imprimir e enviar cópias em papel. Isso é particularmente verdadeiro para as informações que mudam regularmente. A Internet pode economizar seu dinheiro facilitando a comunicação. Quer isso signifique que ofertas podem ser amplamente distribuídas e rapidamente respondidas, quer signifique que os clientes podem comunicar-se diretamente com um atacadista ou o próprio fabricante, eliminando intermediários, o resultado é o mesmo. Os preços podem cair ou os lucros podem subir.

Substituir serviços que custam dinheiro para manter por uma versão eletrônica pode cortar custos. Um exemplo corajoso é o da Egghead.com. Eles escolheram fechar sua cadeia de lojas de computador e concentrar-se em atividades de comércio eletrônico. Embora construir um site significativo de comércio eletrônico obviamente custe dinheiro, uma cadeia de mais de 70 lojas de varejo tem custos contínuos muito mais altos. Substituir um serviço existente é acompanhado de riscos. No mínimo, você perderá clientes que não utilizam a Internet.

A aventura da Egghead.com não funcionou. A empresa fechou suas lojas físicas durante o boom das empresas ponto-com em 1998 e pediram proteção contra a falência durante a queda das empresas ponto-com em 2001.

A centralização pode cortar custos. Se tiver numerosas lojas físicas, você precisa pagar numerosos aluguéis e despesas gerais, os funcionários de todos elas e os custos de manter estoque em cada uma. Um negócio de Internet pode estar em uma localização, mas ser acessível a todos.

Entendendo os riscos e as ameaças

Todos os negócios enfrentam riscos como concorrência, roubo, preferências públicas inconstantes e desastres naturais, entre outros. A lista é interminável. Mas, muitos riscos que empresas de comércio eletrônico correm são menores ou não são relevantes. Esses riscos incluem:

- Crackers;
- Fracasso em atrair negócio suficiente;
- Falha de hardware de computador;

- Falhas de energia, comunicação ou rede;
- Confiança nos serviços de entrega;
- Concorrência grande;
- Erros de software;
- Envolvimento com questões políticas governamentais e impostos;
- Limites de capacidade do sistema.

Crackers

A ameaça mais famosa ao comércio eletrônico provém de usuários maliciosos de computador conhecidos como *crackers*. Todos os negócios correm o risco de tornarem-se alvos de atividades criminosas, mas os negócios de comércio eletrônico de alto perfil com certeza atraíram a atenção de crackers com intenções e capacidades variadas.

Os crackers talvez ataquem pelo desafio, pela notoriedade, para sabotar o site, roubar dinheiro ou ganhar mercadoria ou serviços gratuitamente.

Proteger o site envolve uma combinação de:

- Manter backups de informações importantes;
- Empregar políticas que atraiam pessoas honestas e mantê-las leais – os ataques mais perigosos podem vir de dentro;
- Tomar precauções baseadas em software, como instalar softwares de segurança e mantê-los atualizados;
- Treinar pessoal para identificar alvos e fraquezas;
- Auditorias e registros em logs para detectar invasões ou tentativas de invasões.

Os ataques mais bem-sucedidos em sistemas de computador tiram proveito de fraquezas bem-conhecidas como senhas adivinhadas com facilidade, configurações comuns feitas erroneamente e versões de software antigas. Algumas precauções sensatas podem desviar ataques de não-especialistas e assegurar que você tenha um backup se acontecer o pior.

Fracasso em atrair negócio suficiente

Embora os ataques por crackers sejam amplamente temidos, a maioria das falhas de comércio eletrônico se relaciona a fatores econômicos tradicionais. É muito caro construir e vender um site de comércio eletrônico importante. As empresas estão dispostas a perder dinheiro a curto prazo, com base em suposições de que depois que a marca se estabelecer no mercado, a quantidade de clientes e os lucros aumentarão.

A queda das empresas ponto-com levou muitas empresas à falência enquanto capital de risco precisava suportar a perda de varejistas. A fila de fracassos de alto nível inclui o Boo.com europeu, que consumiu todo o capital aplicado e passou para outras mãos depois de torrar US$120 milhões em seis meses. Não que o Boo não tenha vendido nada; o fato é que acabou gastando indiscutivelmente muito mais do que vendeu.

Falha de hardware de computador

É quase desnecessário dizer que se o negócio conta com um Web site, a falha de uma parte crítica de um de seus computadores terá um impacto.

Os Web sites ocupados ou cruciais justificam ter diversos sistemas redundantes para que a falha de um sistema não afete a operação do sistema inteiro. Como com todas as ameaças, você precisa

determinar se a chance de perder seu Web site por um dia esperando peças ou consertos justifica a despesa de equipamento redundante.

Muitas máquinas que utilizam Apache, PHP e MySQL são razoavelmente fáceis de configurar, e utilizando a replicação do MySQL, também são fáceis de manter em sincronia, mas elas aumentam bastante seu custo de hardware, host e infra-estrutura.

Falhas de energia, de comunicação, de rede ou de entrega

Se contar com a Internet, você está contando com uma complexa malha de provedores de serviço. Se sua conexão com o resto do mundo falhar, você não pode fazer quase nada além de esperar seu fornecedor reintegrar o serviço. O mesmo acontece com interrupções de serviço de energia e greves ou outras interrupções de sua empresa de entregas.

Dependendo de seu orçamento, talvez você escolha manter vários serviços de fornecedores diferentes. Isso custará mais a você, mas significará que, se um dos fornecedores falhar, você ainda terá outro. Faltas de energia breves podem ser sobrepujadas com investimentos em no-breaks.

Grande concorrência

Se estiver abrindo uma loja de vendas a varejo na esquina de uma rua, provavelmente você será capaz de fazer uma pesquisa muito exata do cenário da concorrência. Seus concorrentes serão principalmente empresas que vendem produtos semelhantes nas imediações. Novos concorrentes abrirão ocasionalmente. Com o comércio eletrônico, o terreno é menos certo.

Dependendo de custos de remessa, seus concorrentes poderiam estar em qualquer lugar do mundo e sujeitos a diferentes flutuações cambiais e custos de trabalho. A Internet é ferozmente competitiva e está desenvolvendo-se rapidamente. Se você estiver competindo em uma categoria popular, novos competidores podem aparecer a cada dia.

Há pouco que se possa fazer para eliminar o risco de concorrência, mas, permanecendo a par de desenvolvimentos, você pode assegurar que seu empreendimento continue competitivo.

Erros de software

Quando seu negócio conta com software, você está vulnerável a erros desse tipo.

Você pode reduzir a probabilidade de erros críticos selecionando software que seja confiável, permitindo tempo suficiente para testar depois que mudar partes do sistema, tendo um processo de teste formal e não permitindo que sejam feitas alterações no sistema ao vivo sem fazer teste em outra parte primeiro.

Você pode reduzir a gravidade de resultados tendo backups atualizados de todos os dados, mantendo configurações conhecidas de software que funcionam ao fazer uma alteração e monitorando a operação do sistema para rapidamente detectar problemas.

Evolução de políticas governamentais e impostos

Dependendo de onde você vive, a legislação relacionada a negócios baseados na Internet talvez seja inexistente ou imatura. É improvável que isso dure. Alguns modelos de negócio talvez sejam ameaçados, regulamentados ou talvez sejam eliminados por legislação futura. Os impostos talvez sejam adicionados.

Você não pode evitar essas questões. A única maneira de lidar com elas é manter-se atualizado com o que está acontecendo e manter seu site de acordo com a legislação. Talvez você queira considerar a possibilidade de se unir a qualquer grupo de lobby apropriado à medida que as questões surgirem.

Limites da capacidade do sistema

Algo a lembrar ao projetar seu sistema é o crescimento. Seu sistema se tornará, esperamos, cada vez mais ocupado. Ele deve ser projetado de maneira a poder crescer em escala de acordo com o aumento da demanda.

Para crescimento limitado, você pode aumentar capacidade simplesmente comprando hardware mais rápido. Há um limite para a rapidez do computador que você pode comprar. O software que você utiliza é escrito de tal modo que depois de alcançar esse ponto, você pode separar partes dele para dividir a carga entre os múltiplos sistemas? Seu banco de dados pode tratar diversas solicitações concorrentes provenientes de máquinas diferentes?

Poucos sistemas superam facilmente o crescimento maciço, mas se você projetar com a escalabilidade em mente, deve ser capaz de identificar e eliminar gargalos à medida que a base de clientes cresce.

Adotando uma estratégia

Algumas pessoas acreditam que a Internet muda muito rápido para permitir planejamento efetivo. Argumentaríamos que é essa grande alterabilidade que torna o planejamento crucial. Sem definir objetivos e decidir uma estratégia, você reagirá às alterações à medida que elas ocorrem, em vez de ser capaz de agir em antecipação às alterações.

Tendo examinado alguns objetivos típicos para um Web site comercial, e algumas das suas principais ameaças, acreditamos que você já começou a desenvolver suas próprias estratégias.

Sua estratégia precisará identificar um modelo de negócio. Normalmente o modelo será algo que foi demonstrado como funcional em outro lugar, mas é às vezes uma nova idéia em que você acredita. Você adaptará seu modelo de negócio existente à Web, simulará um competidor existente ou criará agressivamente um serviço pioneiro?

A seguir

No próximo capítulo, examinaremos especificamente a segurança para comércio eletrônico, fornecendo uma visão geral das questões de segurança, os riscos e as técnicas disponíveis.

15

Questões de segurança de comércio eletrônico

ESTE CAPÍTULO DISCUTE O PAPEL DA SEGURANÇA no comércio eletrônico. Discutiremos quem poderia estar interessado nas suas informações e como eles talvez tentem obtê-las, os princípios envolvidos em criar uma política para evitar esses tipos de problemas e algumas tecnologias disponíveis para salvaguardar a segurança de um Web site incluindo criptografia, autenticação e monitoramento.

Os tópicos-chave deste capítulo incluem:

- Qual é a importância de suas informações?;
- Ameaças à segurança;
- Criação de uma política de segurança;
- Equilíbrio entre usabilidade, desempenho, custo e segurança;
- Princípios de autenticação;
- Utilização de autenticação;
- Princípios básicos de criptografia;
- Criptografia de chave privada;
- Criptografia de chave pública;
- Assinaturas digitais;
- Certificados digitais;
- Servidores Web seguros;
- Auditoria e registro em log;
- Firewalls;
- Backup dos dados;
- Segurança física.

Qual é a importância de suas informações?

Ao considerar segurança, a primeira avaliação a ser feita é a importância do que você está protegendo. É preciso considerar sua importância tanto para você como para crackers em potencial.

Talvez seja tentador acreditar que o nível mais alto possível de segurança seja requerido para todos os sites todas as vezes, mas a proteção tem um custo. Antes de decidir quanto esforço ou dinheiro investir na segurança, você precisa decidir quanto valem suas informações.

O valor das informações armazenadas no computador de um usuário que usa o computador como passatempo, de um negócio, de um banco e de uma organização militar obviamente varia. O esforço que um invasor dedicaria para obter acesso a essas informações varia de maneira semelhante. Quanto o conteúdo de suas máquinas seria atraente para um visitante mal-intencionado?

Os usuários de computador por passatempo provavelmente terão tempo limitado para aprender ou trabalhar no sentido de proteger seus sistemas. Dado que essas informações armazenadas em suas máquinas provavelmente serão de valor limitado para qualquer pessoa que não seja seu proprietário, os ataques serão provavelmente raros e envolverão esforço limitado. Entretanto, todos os usuários de computador de rede devem tomar precauções sensatas. Mesmo o computador com os dados menos interessantes ainda tem um poder de atração significativo como uma base de lançamento de ataques anônimos em outros sistemas ou como veículo de reprodução de vírus.

Os computadores militares são um alvo óbvio tanto para indivíduos como para governos estrangeiros. Uma vez que os perpetradores de ataques a instituições governamentais talvez tenham recursos extensos, seria inteligente investir em pessoal e em outros recursos para assegurar que todas as precauções práticas sejam tomadas nesse domínio.

Se você for responsável por um site de comércio eletrônico, sua capacidade de atrair crackers presumivelmente se encaixa em algum lugar entre esses dois extremos.

Ameaças de segurança

O que está em risco no site? Que ameaças existem por aí?

Discutimos algumas ameaças para um negócio de comércio eletrônico no Capítulo 14. Muitas dessas se referem à segurança.

Dependendo do Web site, as ameaças à segurança podem incluir:

- Exposição de dados confidenciais;
- Perda ou destruição de dados;
- Adulteração de dados;
- Negação de serviço;
- Erros no software;
- Repúdio.

Vamos examinar individualmente cada uma dessas ameaças.

Exposição de dados confidenciais

Os dados armazenados em computadores, ou sendo transmitidos para ou a partir de computadores, podem ser confidenciais. Talvez sejam informações que somente certas pessoas pretendam ver, como listas de preço por atacado. Talvez sejam informações confidenciais fornecidas por um cliente, como sua senha, detalhes de contato e número de cartão de crédito.

Esperamos que você não esteja armazenando no servidor Web informações que não pretende que ninguém veja. Um servidor Web é o lugar errado para informações secretas. Se estivesse armazenando seus registros de folha de pagamento ou seu plano para dominar o mundo em um computador, você seria inteligente utilizando um computador diferente do seu servidor Web. O servidor Web é inerentemente uma máquina publicamente acessível e só deve conter informações que precisam ser fornecidas ao público ou que foram recentemente coletadas do público.

Para reduzir o risco de exposição, você precisa limitar os métodos pelos quais as informações podem ser acessadas e limitar as pessoas que podem acessá-las. Isso envolve projetar com a segurança em mente, configurando o servidor e o software de forma adequada, programando com cuidado, testando completamente, removendo serviços desnecessários do servidor Web e requerendo autenticação.

Projete, configure, codifique e teste de forma cuidadosa para reduzir o risco de um ataque criminoso bem-sucedido e, igualmente importante, para reduzir a chance de que um erro deixe suas informações abertas à exposição acidental.

Remova serviços desnecessários do servidor Web para diminuir o número de pontos fracos potenciais. Talvez todos os serviços que você está mantendo tenham vulnerabilidades. Cada um precisa estar atualizado para assegurar que vulnerabilidades conhecidas não estejam presentes. Os serviços que você não utiliza talvez sejam mais perigosos. Se você nunca utiliza o comando `rcp`, por que instalou-o?[1] Se você diz ao instalador que sua máquina é um host de rede, as distribuições importantes Linux e Windows NT instalam vários serviços de que você não precisa e que deve remover.

Autenticação significa pedir às pessoas para provarem sua identidade. Quando o sistema souber quem está fazendo uma solicitação, ele pode decidir se essa pessoa tem permissão de acesso. Há vários métodos de autenticação possíveis, mas somente duas formas comumente utilizadas – senhas e assinaturas digitais. Conversaremos um pouco mais sobre ambas mais tarde.

A CD Universe ofereceu um bom exemplo do custo, tanto em dólares como em reputação, ao permitir a exposição de informações confidenciais. No final de 1999, um cracker autodenominado Maxus contatou, segundo os relatos, a CD Universe, alegando ter 300.000 números de cartão de crédito roubados do site dessa empresa. Ele queria um resgate de U$100.000 dólares do site para destruir os números. A empresa recusou e se viu na constrangedora situação de virar assunto de capa dos principais jornais do país quando Maxus distribuiu os números para outros abusarem.

Os dados também estão em risco de exposição enquanto atravessam uma rede. Embora as redes de TCP/IP tenham muitos bons recursos que os tornam padrão *de facto* para conectar diversas redes conjuntamente como a Internet, a segurança não é um deles. O TCP/IP funciona cortando seus dados em pacotes e então encaminhando esses pacotes de máquina para máquina até que alcancem seu destino. Isso significa que seus dados estão passando por numerosas máquinas no caminho, como ilustra a Figura 15.1. Qualquer uma dessas máquinas poderia visualizar seus dados quando estes passam por elas.

Figura 15.1 Transmitir informações via Internet envia as informações por meio de vários hosts potencialmente não-confiáveis.

Para ver o caminho que os dados tomam a partir de você para uma máquina particular, você pode utilizar o comando `traceroute` (em uma máquina Unix). Esse comando fornecerá os endereços de máquinas pelos quais passam seus dados para alcançar esse host. Para um host no seu próprio país, os dados possivelmente passariam por 10 máquinas diferentes. Para uma máquina internacional, pode haver mais de 20 intermediários. Se sua organização tiver uma grande rede complexa, os dados talvez passem por cinco máquinas antes mesmo de deixar o edifício.

[1] Mesmo se você atualmente usa `rcp`, deve removê-lo e usar `scp` (cópia segura) no lugar.

Para proteger informações confidenciais, você pode encriptá-las antes de serem enviadas por uma rede e decriptá-las na outra extremidade. Os servidores Web freqüentemente utilizam Secure Sockets Layer (SSL), desenvolvido pela Netscape, para realizar isso enquanto os dados viajam entre servidores Web e navegadores. Essa é uma maneira relativamente barata e que exige pouco esforço para tornar seguras as transmissões, mas como o servidor precisa encriptar e decriptar dados em vez de simplesmente enviá-los e recebê-los, o número de visitantes por segundo que uma máquina pode servir cai de modo significativo.

Perda ou destruição de dados

A perda de dados pode ser mais cara do que sua revelação. Se você gastou meses construindo seu site, coletando dados de usuário e pedidos, quanto teria custado, em tempo, reputação e dólares perder todas essas informações? Se não tiver nenhum backup dos dados, você precisará reescrever o Web site depressa e desde o zero. Você também teria clientes insatisfeitos e fraudadores dizendo que encomendaram algo que nunca chegou.

É *possível* que crackers dominem seu sistema e formatem sua unidade de disco. É relativamente *provável* que um programador ou administrador negligente exclua algo de forma acidental e *é quase certo* que você ocasionalmente perderá uma unidade de disco rígido. As unidades de disco rígido giram milhares de vezes por minuto, e, às vezes, elas falham. A lei de Murphy diria que o mais importante falhará, provavelmente muito depois de você ter feito backup pela última vez.

Você pode tomar várias medidas para reduzir a chance de perda de dados. Proteja seus servidores contra crackers. Mantenha a um mínimo o número de funcionários com acesso a sua máquina. Empregue somente pessoas cuidadosas e competentes. Compre unidades de boa qualidade. Utilize RAID para que várias unidades possam atuar como uma unidade mais confiável e mais rápida.

Independente da causa, há somente uma proteção real contra perda de dados – backups. Fazer backup dos dados não é engenharia espacial. Pelo contrário, é tedioso, maçante e, tomara, inútil, mas é vital. Certifique-se de que foram feitos backups dos seus dados regularmente e certifique-se de que testou seu procedimento de backup para ter certeza de que pode realizar uma recuperação dos dados. Certifique-se de que seus backups estão armazenados longe de seus computadores. Embora seja improvável que suas instalações se queimem ou sofram algum outro destino catastrófico, armazenar um backup fora do escritório é uma política de segurança relativamente barata.

Adulteração de dados

Embora a perda de dados possa causar danos, sua adulteração poderia ser pior. E se alguém obtivesse acesso a seu sistema e adulterasse os arquivos? Embora a exclusão em grande escala provavelmente seja notada e possa ser reparada a partir de seu backup, quanto tempo você levaria para notar uma adulteração?

A adulteração poderia incluir alterações nos arquivos de dados ou arquivos executáveis. A motivação de um cracker para alterar um arquivo de dados talvez seja subornar seu site ou obter benefícios fraudulentos. Substituir arquivos executáveis por versões sabotadas também pode fornecer a um cracker, que ganhou acesso uma vez, uma porta dos fundos secreta para futuras visitas.

Você pode proteger dados contra adulteração enquanto esses dados viajam na rede criando uma assinatura eletrônica. Isso não impede que alguém modifique os dados, mas se o destinatário verificar que a assinatura não corresponde quando receber o arquivo, ele saberá que o arquivo foi adulterado. Se os dados estiverem sendo encriptados para protegê-los contra visualização não-autorizada, isso também dificultará bastante a adulteração no percurso sem detecção.

Proteger os arquivos armazenados no servidor contra adulteração exige utilizar os recursos de permissão de arquivo que o sistema operacional fornece e proteger o sistema contra acesso não-autorizado. Utilizando as permissões de arquivo, os usuários podem ser autorizados a utilizar o sistema, mas não recebem o direito de modificar arquivos de sistema e outros arquivos dos usuários.

A falta de um sistema adequado de permissões é uma das razões pelas quais o Windows 95 e o 98 não são sistemas operacionais adequados de servidor.

Detectar adulteração pode ser difícil. Se em algum ponto você percebe que a segurança do sistema foi corrompida, como saberá se arquivos importantes foram modificados? Alguns arquivos, como os arquivos de dados que armazenam bancos de dados, são concebidos para serem alterados com o decorrer do tempo. Muitos outros são concebidos para permanecerem os mesmos desde o momento em que você os instala, a menos que você deliberadamente os atualize. A adulteração tanto de programas como de dados pode ser insidiosa, mas embora programas possam ser reinstalados em caso de suspeita de adulteração, você não pode saber qual versão dos dados está "limpa".

Um software de avaliação de integridade de arquivo, como o Tripwire, registra informações sobre arquivos importantes em um estado seguro conhecido, provavelmente imediatamente depois da instalação e pode ser utilizado mais tarde para verificar se arquivos estão inalterados. Você pode fazer download das versões gratuitas comerciais ou condicionais em:

http://www.tripwire.com

Negação de serviço

Uma das ameaças mais difíceis de prevenir é a negação de serviço. A *negação de serviço* (Denial of Service – DoS) ocorre quando as ações de alguém dificultam ou impossibilitam os usuários de acessarem um serviço ou quando retardam seu acesso a um serviço de tempo crítico.

No início de 2000, houve uma famigerada investida de ataques do tipo *Distributed Denial of Service (DDoS)* contra Web sites famosos. Os alvos incluíram Yahoo!, eBay, Amazon, E-Trade e Buy.com. Sites como esses são acostumados a níveis de tráfego que a maioria de nós nem poderia sonhar, mas são ainda vulneráveis a serem desligados durante horas por um ataque DoS. Embora os crackers geralmente tenham pouco a ganhar ao desligar um Web site, o proprietário talvez perca dinheiro, tempo e reputação.

Alguns sites têm horas em que esperam realizar mais negócios. Sites de reservas on-line têm uma demanda muito grande antes de grandes eventos esportivos. Uma maneira que os crackers tentaram lucrar com ataques DDoS em 2004 foi extorquindo dinheiro de sites de reservas on-line com a ameaça de atacar durante essas horas de pico.

Uma das razões de ser tão difícil de prevenir esses ataques é que há uma enorme variedade de maneiras de executá-los. Os métodos poderiam incluir instalar um programa em uma máquina-alvo que utiliza a maior parte do tempo de processador do sistema, inundação inversa ou utilizar uma das ferramentas automatizadas. Uma *inundação inversa* (*reverse spam*) envolve alguém inundar a Internet com correio apócrifo tendo a vítima como remetente. Dessa maneira, a vítima terá de lidar com milhares de respostas enfurecidas.

Existem ferramentas automatizadas para carregar ataques de DoS distribuídos contra um alvo. Sem precisar de muito conhecimento, alguém pode varrer uma grande quantidade de máquinas em busca de vulnerabilidades conhecidas, comprometer uma máquina e instalar a ferramenta. Como o processo é automatizado, um invasor pode instalar a ferramenta em um único host em menos de cinco segundos. Quando máquinas suficientes foram cooptadas, todas são instruídas a inundar o alvo com tráfego de rede.

Em geral, é difícil prevenir ataques de DoS. Com uma pequena pesquisa, você pode localizar as portas padrão utilizadas pelas ferramentas DDoS comuns e fechá-las. Seu roteador talvez forneça mecanismos como limitar a porcentagem de tráfego que utiliza protocolos particulares como ICMP. Detectar hosts na rede sendo utilizados para atacar outros é mais fácil do que proteger a máquina contra o ataque. Se pudéssemos confiar em cada administrador de rede para monitorar vigilantemente nossa própria rede, o DDoS não seria um problema dessa proporção.

Como há tantos métodos de ataque possíveis, a única defesa realmente efetiva é monitorar o comportamento de tráfego normal e ter um grupo de especialistas disponíveis para tomar contramedidas quando ocorrerem anormalidades.

Erros em software

É possível que o software que você comprou, obteve ou escreveu tenha erros sérios. Dados os curtos períodos de desenvolvimento normalmente permitidos para projetos Web, é muito provável que esse software tenha alguns erros. Qualquer negócio que seja altamente baseado em processos computadorizados está vulnerável a software falho.

Os erros no software podem levar a todo tipo de comportamento imprevisível incluindo a inviabilização do serviço, brechas de segurança, perdas financeiras e mau serviço para os clientes.

As causas comuns de erros que você pode procurar incluem más especificações, suposições incorretas feitas por desenvolvedores e teste inadequado.

Más especificações

Quanto mais esparsa ou ambígua for a documentação do projeto, maior a probabilidade de você terminar com erros no produto final. Embora talvez pareça supérfluo especificar isso quando o cartão de crédito de um cliente é recusado, o pedido não deve ser enviado para o cliente, pelo menos um site de grande orçamento teve esse bug. Quanto menos experiência seus desenvolvedores têm com o tipo de sistema em que estão trabalhando, mais exata precisa ser sua especificação.

Suposições feitas por desenvolvedores

Os projetistas e programadores de um sistema precisam fazer muitas suposições. Provavelmente, esperamos, eles documentarão suas suposições e normalmente estarão corretos. Embora às vezes as pessoas façam más suposições. Essas talvez incluam suposições de que dados de entrada serão válidos, não incluirão caracteres incomuns nem serão menores do que um determinado tamanho. Além disso, poderia incluir suposições sobre sincronização como a probabilidade de duas ações conflitantes ocorrendo ao mesmo tempo ou que uma tarefa complexa de processamento sempre levará mais tempo do que uma tarefa simples.

As suposições como essas podem não causar problemas porque são normalmente verdadeiras. Mas um cracker poderia tirar proveito de um estouro de buffer porque um programador assumiu um comprimento máximo para os dados de entrada; ou um usuário legítimo poderia ficar confuso com mensagens de erro e sair porque não ocorreu aos desenvolvedores que o nome de uma pessoa poderia ter um apóstrofo. Esses tipos de erros podem ser localizados e corrigidos com uma combinação de um bom teste e uma revisão detalhada do código.

Historicamente, as fraquezas de sistema operacional ou de nível de aplicação exploradas por crackers normalmente se relacionam com estouros de buffer ou condições de concorrência.

Teste pobre

Raramente é possível testar todas as condições de entrada possíveis, em todos os tipos de hardware possíveis, executando todos os possíveis sistemas operacionais com todas as possíveis configurações de usuário. Isso é ainda mais verdadeiro que o usual em sistemas baseados na Web.

Necessita-se de um plano de teste bem projetado que teste todas as funções do software em uma amostra representativa de tipos comuns de máquina. Um conjunto bem-planejado de testes deve testar cada linha de código no projeto pelo menos uma vez. Idealmente, esse conjunto de teste deve ser automatizado para que possa ser executado nas máquinas de teste selecionadas com pouco esforço.

O maior problema com os testes é que eles são inglórios e repetitivos. Embora algumas pessoas gostem de quebrar coisas, poucas pessoas gostam de quebrar a mesma coisa repetidamente. É importante que outras pessoas que não sejam os desenvolvedores originais estejam envolvidos no teste. Um dos objetivos importantes de testar é descobrir suposições defeituosas feitas pelos desenvolvedores. Uma pessoa nova muito provavelmente tem suposições diferentes. Além disso, os profissionais raramente estão propensos a encontrar defeitos no seu próprio trabalho.

Repúdio

O risco final que consideraremos é o repúdio. O *repúdio* ocorre quando uma parte envolvida em uma transação nega ter participado dela. Os exemplos de comércio eletrônico talvez incluam uma pessoa que encomenda mercadoria de um Web site e então nega ter autorizado a taxa no cartão de crédito, ou uma pessoa que concorda com algo no e-mail e então afirma que uma outra pessoa forjou o e-mail.

Idealmente, as transações financeiras devem fornecer a tranqüilidade do não-repúdio às duas partes. Nenhuma parte poderia negar sua participação em uma transação, ou, mais precisamente, as duas partes poderiam conclusivamente provar as ações da outra para uma terceira parte, como em um tribunal. Na prática, isso raramente acontece.

A autenticação fornece alguma segurança sobre com quem você está lidando. Se emitida por uma organização de confiança, os certificados digitais de autenticação podem fornecer maior confiança.

As mensagens enviadas por cada parte também precisam ser à prova de falsificação. Não há muito valor em poder demonstrar que a Corp Pty Ltd enviou-lhe uma mensagem se você também não puder demonstrar que o que recebeu foi exatamente o que eles enviaram. Como mencionado anteriormente, assinar ou encriptar mensagens torna difícil alterá-las sorrateiramente.

Para as transações entre partes com um relacionamento em progresso, os certificados digitais conjuntamente com comunicações encriptadas ou assinadas são uma maneira efetiva de limitar o repúdio. Para transações do tipo *one-off*, isto é, de uma única vez, como o contato inicial entre um Web site de comércio eletrônico e um estranho usando um cartão de crédito, os certificados digitais não são tão práticos.

Uma empresa de comércio eletrônico deve estar disposta a ceder prova de sua identidade e algumas centenas de dólares para uma autoridade de certificação como a VeriSign (http://www.verisign.com/) ou a Thawte (http://www.thawte.com/) para garantir aos visitantes a boa-fé da empresa. Essa mesma empresa estaria disposta a rejeitar cada cliente que não estivesse disposto a fazer o mesmo a fim de provar sua identidade? Para transações pequenas, os comerciantes geralmente estão dispostos a aceitar um certo nível de fraude ou risco de repúdio em vez de rejeitar o negócio.

Equilibrando usabilidade, desempenho, custo e segurança

Pela sua própria natureza, a Web é arriscada. A Web foi projetada para permitir que numerosos usuários anônimos solicitem serviços provenientes de máquinas servidoras, que poderiam ser as máquinas que você está administrando. A maioria dessas solicitações será perfeitamente legítima para páginas Web, mas conectar suas máquinas à Internet permitirá que as pessoas tentem outros tipos de conexões.

Embora possa ser tentador assumir que o nível mais alto possível de segurança seja apropriado, esse raramente é o caso. Se quisesse estar realmente seguro, você manteria todos os seus computadores desligados, desconectados de todas as redes, em um cofre trancado. A fim de fazer seus computadores disponíveis e utilizáveis, algum relaxamento da segurança é necessário.

Há uma relação de troca entre segurança, usabilidade, custo e desempenho. Tornar um serviço mais seguro pode reduzir usabilidade, por exemplo, limitando o que as pessoas podem fazer ou exigindo que elas se identifiquem. Aumentar segurança também pode reduzir o nível de desempenho de suas máquinas. Executar software para tornar seu sistema mais seguro – como criptografia, sistemas de detecção de invasão, scanners antivírus e extenso registro em log – consome recursos do sistema. É necessário muito mais poder de processamento para atender a uma sessão encriptada, como uma conexão de SSL para um Web site, que para atender a uma normal. Essas perdas no desempenho podem ser contornadas gastando mais dinheiro em máquinas mais rápidas ou hardware especificamente projetado para criptografia.

Você pode considerar desempenho, usabilidade, custo e segurança como objetivos concorrentes. Você deve examinar o preço exigido e tomar decisões sensatas para chegar a um acordo. Depen-

dendo do valor das informações, do orçamento, do número de visitantes que você espera atender e dos obstáculos que você imagina que usuários legítimos estarão dispostos a tolerar, é possível chegar a uma solução conciliatória.

Criando uma política de segurança

Uma política de segurança é um documento que descreve:

- A filosofia geral com relação à segurança na sua organização;
- O que deve ser protegido – software, hardware, dados;
- Quem é responsável pela proteção desses itens;
- Padrões para segurança e medições, que avaliam o quanto esses padrões estão sendo obedecidos.

Uma boa diretriz para escrever sua política de segurança é que ela é semelhante a escrever um conjunto de requisitos funcionais para software. A política não deve falar sobre implementações específicas ou soluções, mas em vez disso sobre os objetivos e requisitos de segurança no seu ambiente. Essa política não deve precisar ser atualizada com muita freqüência.

Você deve manter um documento separado que configure diretrizes para a maneira como os requisitos da política de segurança são obedecidos em um ambiente particular. Você pode ter diretrizes diferentes para diferentes partes de sua organização. Isso é mais parecido com um documento de design ou um manual de procedimento que documente o que é realmente feito a fim de assegurar o nível de segurança exigido.

Princípios de autenticação

A *autenticação* tenta provar que alguém é realmente quem afirma ser. Há muitas maneiras possíveis de fornecer autenticação, mas como com muitas medidas de segurança, os métodos mais seguros são mais incômodos de utilizar.

As técnicas de autenticação incluem senhas, assinaturas digitais, medidas biométricas como varreduras de impressão digital e medidas envolvendo hardware como cartões inteligentes. Somente duas técnicas estão em utilização comum na Web: senhas e assinaturas digitais.

As medidas biométricas e a maioria de soluções de hardware envolvem dispositivos de entrada especiais e limitaria usuários autorizados a máquinas específicas a que esses dispositivos estão anexados. Isso talvez seja aceitável ou mesmo desejável, para acessar sistemas internos de uma organização, mas tira muita da vantagem de tornar um sistema disponível pela Web.

As senhas são simples de implementar, de utilizar e não exigem nenhum dispositivo de entrada especial. As senhas fornecem algum nível de autenticação, mas, por si só, poderiam não ser adequadas para sistemas de alta segurança.

Uma senha é um conceito simples. Você e o sistema conhecem sua senha. Se um visitante afirma ser você e sabe sua senha, o sistema tem razão de acreditar que ele é você. Contanto que ninguém mais saiba ou possa adivinhar a senha, esta é segura. As senhas têm por si só várias fraquezas potenciais e não fornecem autenticação forte.

Muitas senhas são facilmente adivinhadas. Se você deixar seus usuários escolherem suas próprias senhas, aproximadamente 50% deles escolherão uma senha facilmente adivinhável. As senhas comuns que se ajustam nessa descrição incluem palavras de dicionário ou o nome de usuário para a conta. À custa da usabilidade, você pode forçar usuários a incluírem números ou pontuação em suas senhas, mas isso fará com que alguns usuários tenham dificuldade de lembrar suas senhas. Educar usuários para escolherem melhor as senhas pode ajudar, mas mesmo quando educados, aproximadamente 25% dos usuários ainda escolherão uma senha facilmente adivinhável. Você poderia impor políticas de senha que impeçam que os usuários escolham combinações facilmente adivinháveis verificando novas senhas contra um dicionário ou exigindo alguns números ou símbolos de pontua-

ção ou uma combinação de letras minúsculas e maiúsculas. Um perigo é que regras estritas de senha levarão a senhas que muitos usuários legítimos não serão capazes de lembrar.

A dificuldade de lembrar senhas aumenta a probabilidade de que usuários farão algo não seguro como escrever "username fred password rover" em uma nota adesiva colada no monitor da sua máquina. Os usuários precisam ser educados a não anotar suas senhas ou fazer outras coisas ridículas como fornecê-las a pessoas pelo telefone que afirmam estar trabalhando no sistema.

As senhas também podem ser capturadas eletronicamente. Executando um programa para capturar pressionamentos de tecla em um terminal ou utilizando um *packet sniffer* para capturar tráfego de rede, crackers podem – e realmente conseguem – capturar pares utilizáveis de nomes de login e senhas. Você pode limitar as oportunidades de capturar senhas encriptando o tráfego da rede.

Para todos seus defeitos potenciais, as senhas são uma maneira relativamente eficiente e simples de autenticar usuários. As senhas fornecem um nível de segredo que talvez não seja apropriado para segurança nacional, mas é ideal para verificar o status de entrega de um pedido do cliente.

Utilizando autenticação

Os mecanismos de autenticação são construídos para os navegadores mais populares Web e para servidores Web. Os servidores Web talvez requeiram um nome de usuário e senha para as pessoas solicitando arquivos e diretórios particulares no servidor.

Quando desafiado a fornecer um nome de login e senha, seu navegador apresentará uma caixa de diálogo parecida com a mostrada na Figura 15.2.

Figura 15.2 Os navegadores Web solicitam aos usuários a autenticação quando eles tentam visitar um diretório restrito em um servidor Web.

Tanto o servidor Web Apache como IIS da Microsoft permitem projetar muito facilmente todo ou parte de um site dessa maneira. Utilizando o PHP ou o MySQL, há muitas outras maneiras de alcançar o mesmo efeito. Utilizar o MySQL é mais rápido que a autenticação predefinida. Utilizando o PHP, podemos fornecer autenticação mais flexível ou apresentar a solicitação de maneira mais atraente.

Veremos alguns exemplos de autenticação no Capítulo 16.

Princípios básicos de criptografia

Um *algoritmo de criptografia* é um processo matemático para transformar informações em uma string de dados aparentemente aleatória.

Os dados com que você inicia são freqüentemente chamados de *texto simples*, embora não seja importante para o processo o que as informações representam – quer elas sejam realmente texto ou algum outro tipo de dados. De maneira semelhante, as informações encriptadas são chamadas de *texto de cifra*, mas raramente o texto de cifra é parecido com texto. A Figura 15.3 mostra o processo

de criptografia como um fluxograma simples. O texto simples é fornecido para um mecanismo de criptografia, que poderia ser um dispositivo mecânico, como uma máquina Enigma da Segunda Guerra Mundial muito tempo atrás, mas atualmente é quase sempre um programa de computador. O mecanismo produz o texto de cifra.

Figura 15.3 A criptografia aceita o texto simples e o transforma em texto de cifra aparentemente aleatório.

Para criar o diretório protegido cujo prompt de autenticação é mostrado na Figura 15.2, utilizamos o tipo de autenticação mais básico do Apache. (Você verá como utilizar isso no próximo capítulo.) Isso encripta senhas antes de armazená-las. Criamos um usuário com a senha `password`. Essa senha foi encriptada e armazenada como `aWDuA3X3H.mc2`. Você pode ver que o texto simples e o texto de cifra não contêm nenhuma semelhança óbvia entre si.

Esse método particular de criptografia não é reversível. Muitas senhas são armazenadas utilizando um algoritmo de criptografia de uma via. A fim de determinar se uma tentativa de inserir uma senha é correta, não precisamos decriptar a senha armazenada. Podemos, em vez disso, encriptar a tentativa e comparar com a versão armazenada.

Muitos, mas nem todos, os processos de criptografia podem ser inversos. O processo inverso é chamado *decriptografia*. A Figura 15.4 mostra um processo de criptografia de duas vias.

Figura 15.4 A criptografia aceita o texto simples e o transforma em texto de cifra aparentemente aleatório. A decriptografia toma o texto de cifra e o transforma outra vez em texto simples.

A criptografia tem quase 4000 anos, mas atingiu sua maioridade na Segunda Guerra Mundial. Seu crescimento desde então tem seguido um padrão semelhante à adoção de redes de computadores, inicialmente sendo usada apenas por corporações militares e financeiras, depois mais amplamente utilizada por empresas que começaram na década de 1970, até por fim tornarem-se onipresentes na década de 1990. Nos últimos anos, a criptografia partiu de um conceito que pessoas comuns só viam em filmes que tratavam da Segunda Guerra Mundial e filmes de espionagem para algo que elas lêem nos jornais e usam todas as vezes que compram com seus navegadores Web.

Muitos algoritmos de criptografia diferentes estão disponíveis. Alguns, como o DES, utilizam um segredo ou chave privada; alguns, como o RSA, utilizam uma chave pública e uma chave privada separada.

Criptografia de chave privada

A criptografia de chave privada se baseia no fato de as pessoas autorizadas conhecerem ou terem acesso a uma chave. Essa chave deve ser mantida em segredo. Se a chave cair nas mãos erradas, pessoas não-autorizadas também podem ler suas mensagens encriptadas. Como mostrado na Figura 15.4, tanto o remetente (que encripta a mensagem) como o destinatário (que decripta a mensagem) têm a mesma chave.

O algoritmo de chave secreta mais amplamente utilizado é o Data Encryption Standard (DES). Esse esquema foi desenvolvido pela IBM na década de 1970 e adotado como o padrão norte-americano para comunicações governamentais não-secretas e comerciais. As velocidades de computação são ordens de magnitudes mais rápidas agora que em 1970 e o DES tornou-se obsoleto desde, pelo menos, 1998.

Outros sistemas bem-conhecidos de chave secreta incluem o RC2, o RC4, o RC5, o DES triplo e o IDEA. O DES triplo é relativamente seguro.[2] Ele utiliza o mesmo algoritmo que o DES, aplicado três vezes com até três chaves diferentes. Uma mensagem simples de texto é encriptada com a chave um, decriptada com a chave dois e então encriptada com a chave três.

Um defeito óbvio de criptografia de chave secreta é que, a fim de enviar uma mensagem segura, você precisa de uma maneira segura para obter a chave secreta para ela. Se você tiver uma maneira segura de entregar uma chave, por que não entregar simplesmente a mensagem dessa maneira?

Felizmente, houve um avanço em 1976, quando Diffie e Hellman publicaram o primeiro esquema de chave pública.

Criptografia de chave pública

A criptografia de chave pública conta com duas chaves diferentes, uma pública e uma privada. Como a Figura 15.5 mostra, a chave pública é utilizada para encriptar mensagens e a chave privada para decriptá-las.

A vantagem desse sistema é que a chave pública, como seu nome sugere, pode ser distribuída publicamente. Qualquer pessoa para quem você forneça sua chave pública pode enviar uma mensagem segura. Contanto que somente você tenha sua chave privada, apenas você poderá decriptar a mensagem.

O algoritmo mais comum de chave pública é o RSA, desenvolvido por Rivest, Shamir e Adelman no MIT e publicado em 1978. O RSA era um sistema patenteado, mas a patente expirou em setembro de 2000.

Figura 15.5 A criptografia de chave pública utiliza chaves separadas para criptografia e decriptografia.

A capacidade de transmitir uma chave pública às claras e não precisar se preocupar com o fato de ela poder ser vista por terceiros é uma vantagem enorme, mas os sistemas de chave secreta ainda estão em utilização comum. Freqüentemente, um sistema híbrido é utilizado. Um sistema de chave pública é utilizado para transmitir a chave para um sistema de chave secreta que será utilizado para o restante de uma sessão de comunicação. Essa complexidade adicional é tolerada porque sistemas de chave secreta são aproximadamente 1.000 vezes mais rápidos que sistemas de chave pública.

Assinaturas digitais

As assinaturas digitais estão relacionadas à criptografia de chave pública, mas invertem o papel de chaves públicas e privadas. Um remetente pode encriptar e assinar digitalmente uma mensagem

[2]De alguma forma paradoxal, o DES triplo é duas vezes mais seguro do que o DES. Se você precisasse de alguma coisa três vezes mais segura, teria de escrever um programa para implementar um algoritmo DES quíntuplo.

com sua chave secreta. Quando a mensagem é recebida, o destinatário pode decriptá-la com a chave pública do remetente. Como o remetente é a única pessoa com acesso à chave secreta, o destinatário pode estar relativamente certo de quem a mensagem veio e de que ela não foi alterada.

As assinaturas digitais podem ser realmente úteis. Elas permitem ao destinatário certificar-se de que a mensagem não foi adulterada e dificultam ao remetente repudiar a, ou negar o envio da, mensagem.

É importante notar que apesar de a mensagem ter sido encriptada, ela pode ser lida por qualquer pessoa que tiver a chave pública. Embora as mesmas técnicas e chaves sejam utilizadas, o propósito de criptografia aqui é impedir interferência e repúdio e não impedir a leitura.

Como a criptografia de chave pública é relativamente lenta para mensagens grandes, outro tipo de algoritmo, chamado *função de hash,* é normalmente utilizado para melhorar a eficiência. A função de hash calcula uma resenha da mensagem ou valor de hash para qualquer mensagem que lhe for fornecida. O importante não é o valor que o algoritmo produz. O importante é que a saída seja determinista, isto é, seja a mesma toda vez que uma entrada particular for utilizada, e seja pequena, e que o algoritmo seja rápido.

As funções de hash mais comuns são o MD5 e o SHA.

Uma função de hash gera uma resenha de mensagem que corresponde a uma mensagem particular. Se tiver uma mensagem e uma resenha de mensagem, você pode verificar se a mensagem não foi adulterada, contanto que tenha certeza de que a resenha não foi adulterada. Para esse fim, a maneira normal de criar uma assinatura digital é criar uma resenha de mensagem para a mensagem inteira utilizando uma função de hash rápida e então encriptar somente a breve resenha utilizando um algoritmo lento de criptografia de chave pública. A assinatura agora pode ser enviada com a mensagem via qualquer método não-seguro normal.

Quando uma mensagem assinada é recebida, ela pode ser verificada. A assinatura é decriptada utilizando a chave pública do remetente. Um valor de hash é gerado para a mensagem utilizando o mesmo método que o remetente utilizou. Se o valor de hash decriptado corresponder ao valor de hash que você gerou, então a mensagem é proveniente do remetente e não foi alterada.

Certificados digitais

É bom ser capaz de verificar se uma mensagem não foi alterada e que uma série de mensagens vem de uma máquina ou usuário particular. Para interações comerciais, seria ainda melhor ser capaz de associar esse usuário ou servidor a uma entidade jurídica real como uma pessoa ou uma empresa.

Um certificado digital combina uma chave pública e detalhes de um indivíduo ou organização em um formato digital assinado. Dado um certificado, você tem a chave pública da outra parte, caso queira enviar uma mensagem encriptada, e tem detalhes dessa parte, que você sabe que não foram alterados.

O problema aqui é que as informações são somente tão confiáveis quanto a pessoa que as assinou. Qualquer pessoa pode gerar e assinar um certificado afirmando ser alguém que ele quiser. Para transações comerciais, seria útil que uma terceira parte de confiança verificasse a identidade de participantes e os detalhes registrados em seus certificados.

Esses terceiros são chamados de *autoridades de certificação* (*Certifying Authorities – CAs*). As autoridades de certificação emitem certificados digitais para indivíduos e empresas sujeitas a verificações de identidade. As duas CAs mais famosas são a VeriSign (http://www.verisign.com/) e a Thawte (http://www.thawte.com/), mas há várias outras autoridades. Tanto a VeriSign como a Thawte pertencem à mesma empresa e há pequena diferença prática entre elas. Algumas autoridades menos conhecidas, como a Equifax Secure (www.equifaxsecure.com), são significativamente mais baratas.

As autoridades assinam um certificado para confirmar que viram uma prova da identidade da pessoa ou da empresa. Vale notar que o certificado não é uma referência ou instrução de validade de crédito. O certificado não garante que você está lidando com alguém respeitável. O que ele realmente significa é que, se você for roubado, terá uma chance muito boa de ter um endereço físico e real e alguém para processar.

Os certificados fornecem uma rede de confiança. Assumindo que escolha confiar na CA, você então pode escolher confiar nas pessoas que escolhem confiar na CA e então confiar nas pessoas em que a CA escolhe confiar.

A utilização mais comum para certificados digitais é fornecer um ar de respeitabilidade para um site de comércio eletrônico. Com um certificado emitido por uma CA bem-conhecida, os navegadores Web podem fazer conexões de SSL com seu site sem abrir caixas de diálogo de aviso. Os servidores Web que permitem conexões de SSL são freqüentemente chamados de *servidores Web seguros*.

Servidores Web seguros

Você pode utilizar o servidor Web Apache, o Microsoft IIS ou todos os outros vários servidores Web gratuitos ou comerciais para comunicação segura com navegadores via Secure Sockets Layer. Utilizar o Apache permite utilizar um sistema operacional do tipo UNIX, que quase certamente será mais confiável, mas é mais difícil de configurar que o IIS. Você pode também, é claro, escolher utilizar o Apache em uma plataforma Windows.

Utilizar o SSL no IIS envolve simplesmente instalar o IIS, gerar um par de chaves e instalar seu certificado. Utilizar SSL no Apache exige instalar três pacotes diferentes: Apache, Mod_SSL e OpenSSL.

Você também pode ter seu bolo e comê-lo comprando Stronghold. O Stronghold é um produto comercial disponível em www.c2.net por aproximadamente US$1.000. Ele é baseado no Apache, mas vem como um binário de auto-instalação pré-configurado com o SSL. Dessa maneira, você obtém a confiabilidade do UNIX junto com um produto de fácil instalação com suporte técnico proveniente do fornecedor.

As instruções de instalação para os dois servidores Web mais populares Web, Apache e IIS, estão no Apêndice A. Você pode começar usando SSL imediatamente gerando seu próprio certificado digital, mas os visitantes do seu site serão advertidos pelos navegadores Web que você assinou seu próprio certificado. A fim de utilizar o SSL efetivamente, você também precisará de um certificado emitido por uma autoridade de certificação.

O processo exato para obter isso varia entre CAs, mas em geral, você precisará provar para uma CA que você é algum tipo de negócio reconhecido juridicamente com um endereço físico e que o negócio em questão possui o nome de domínio relevante.

Você precisa gerar um Certificate Signing Request. O processo para isso variará de servidor para servidor. As instruções estão nos Web sites das CAs. O Stronghold e o IIS fornecem um processo baseado em caixa de diálogo, ao passo que Apache exige digitar comandos. Entretanto, o processo é essencialmente o mesmo para todos os servidores. O resultado final é uma solicitação de assinatura de certificado (certificate signing request – CSR) encriptada. Seu CSR deve se parecer com algo assim:

```
---BEGIN NEW CERTIFICATE REQUEST---
MIIBuwIBAAKBgQCLn1XX8faMHhtzStp9wY6BVTPuEU9bpMmhrb6vgaNZy4dTe6VS
84p7wGepq5CQjfOL4Hjda+g12xzto8uxBkCDO98Xg9q86CY45HZk+q6GyGOLZSOD
8cQHwh1oUP65s5Tz018OFBzpI3bHxfO6aYelWYziDiFKp1BrUdua+pK4SQIVAPLH
SV9FSz8Z7IHOg1Zr5H82oQOlAoGAWSPWyfVXPAF8h2GDb+cf97k44VkHZ+Rxpe8G
ghlfBn9L3ESWUZNOJMfDLlny7dStYU98VTVNekidYuaBsvyEkFrny7NCUmiuaSnX
4UjtFDkNhX9j5YbCRGLmsc865AT54KRu3102/dKHLo6NgFPirijHy99HJ4LRY9Z9
HkXVzswCgYBwBFH2QfK88C6JKW3ah+6cHQ4Deoiltxi627WN5HcQLwkPGn+WtYSZ
jG5tw4tqqogmJ+IP2F/5G6FI2DQP7QDvKNeAU8jXcuijuWo27S2sbhQtXgZRTZvO
jGn89BCOmIHgHQMkI7vz35mx1Skk3VNq3ehwhGCvJlvoeiv2J8X2IQIVAOTRp7zp
En7QlXnXw1s7xXbbuKPO
---END NEW CERTIFICATE REQUEST---
```

Munido de um CSR, a taxa apropriada e a documentação para provar que você existe e tendo verificado se o nome de domínio utilizado está no mesmo nome que a documentação de negócio, você pode inscrever-se para obter um certificado com uma CA.

Quando a CA emitir seu certificado, você precisa armazená-lo em seu sistema e informar a seu servidor Web onde localizá-lo. O certificado final é um arquivo de texto que parece bastante com o CSR mostrado anteriormente.

Auditoria e registro em log

Seu sistema operacional permitirá registrar em log todos os tipos de eventos. Os eventos em que você pode estar interessado de um ponto de vista de segurança incluem erros de rede, acesso a arquivos de dados particulares como arquivos de configuração ou o registro de NT e chamadas a programas como su (utilizado para tornar-se outro usuário, em geral root, em um sistema UNIX).

Os arquivos de log podem ajudar a detectar comportamentos errôneos ou maliciosos à medida que estes ocorrem. Os arquivos de log também podem instruir como um problema ou invasão ocorreu se você os verificar depois de notar os problemas. Há dois problemas principais com arquivos de log: tamanho e veracidade.

Se definir os critérios para detectar e registrar em log problemas no seu nível mais paranóico, você terminará com logs maciços que são muito difíceis de examinar. Para ajudar com grandes arquivos de log, você realmente precisa utilizar uma ferramenta existente ou derivar alguns scripts de auditoria a partir de sua política de segurança para pesquisar nos logs os eventos "interessantes". O processo de auditoria poderia ocorrer em tempo real ou ser feito periodicamente.

Os arquivos de log são vulneráveis a ataques. Se um invasor tiver acesso de root ou acesso de administrador a seu sistema, ele estará livre para alterar arquivos de log a fim de ocultar seus rastros. O Unix fornece recursos para gravar eventos de log em uma máquina separada. Isso significa que um cracker precisaria comprometer pelo menos duas máquinas para cobrir seus rastros. Funcionalidade semelhante é possível no NT, mas não tão facilmente.

Seu administrador de sistema poderia fazer auditorias regulares, mas talvez também quisesse ter uma auditoria externa para verificar periodicamente o comportamento dos administradores.

Firewalls

Os firewalls em redes são projetados para separar a rede do resto do mundo. Da mesma maneira como as paredes corta-fogo (firewalls) em um edifício ou um carro impedem que o fogo se espalhe para outros compartimentos, os firewalls de rede impedem que o caos se espalhe em sua rede.

Um *firewall* é projetado para proteger máquinas na sua rede contra ataque externo. O firewall filtra e nega tráfego que não obedece às regras que você estabeleceu. Ele restringe as atividades das pessoas e máquinas fora do firewall.

Às vezes, um firewall também é utilizado para restringir as atividades dos usuários dentro dele. Um firewall pode restringir protocolos de rede que as pessoas podem utilizar, restringir os hosts a que elas se conectam ou forçá-las a utilizar um servidor proxy para cortar custos de largura de banda.

Um firewall pode ser tanto um dispositivo de hardware, tipo um roteador com regras de filtragem, quanto um programa de software que executa em uma máquina. Em qualquer caso, o firewall precisa de interfaces para duas redes e um conjunto de regras. O firewall monitora todo o tráfego que tenta passar de uma rede para outra. Se o tráfego obedecer às regras, ele é roteado por meio da outra rede; caso contrário, é interrompido ou rejeitado.

Os pacotes podem ser filtrados por seu tipo, endereço de origem, endereço de destino ou informações de porta. Alguns pacotes serão meramente descartados enquanto certos eventos poderiam desencadear entradas de log ou alarmes.

Fazendo backup de dados

Você não pode subestimar a importância dos backups em qualquer plano de recuperação a partir de um desastre. É possível fazer um seguro e/ou substituir o hardware e as instalações físicas e também

é possível hospedar os sites em outra parte, mas se seu software Web desenvolvido e personalizado for perdido, nenhuma empresa de seguro pode substituí-lo para você.

Você precisa fazer backup de todos os componentes de seu Web site – páginas estáticas, scripts e bancos de dados – regularmente. A freqüência com que você faz isso depende do quanto seu site é dinâmico. Se ele for todo estático, você pode obter êxito fazendo o backup dele quando for alterado. Entretanto, conversamos sobre tipos de sites neste livro que se alteram com muita freqüência, em especial se você estiver recebendo pedidos on-line.

A maioria dos sites de um tamanho razoável precisará ser hospedado em um servidor com RAID (Redundant Array of Inexpensive Disks), que pode suportar espelhamento. Isso abrange a situação em que você talvez tenha uma falha de disco rígido. Considere, porém, o que poderia acontecer em uma situação em que algo acontece ao conjunto inteiro, máquina ou prédio.

Você deve realizar backups separados com uma freqüência correspondente ao volume de atualização. Esses backups devem ser armazenados em mídia separada e preferivelmente em uma localização segura e separada, em caso de fogo, roubo ou desastres naturais.

Existem muitos recursos sobre backup e recuperação. Nos concentraremos em como você pode fazer backup de um site construído com PHP e um banco de dados de MySQL.

Fazendo backup de arquivos gerais

O backup de arquivos HTML, PHP, imagens e de outros arquivos não-banco de dados pode ser feito de modo relativamente simples na maioria dos sistemas utilizando software de backup.

O utilitário mais amplamente utilizado dos disponíveis gratuitamente é o AMANDA, o Advanced Maryland Automated Network Disk Archiver, desenvolvido pela University of Maryland. Ele vem com muitas distribuições de UNIX e também pode ser utilizado para fazer backup de máquinas Windows via SAMBA. Você pode ler mais sobre AMANDA em:

http://www.amanda.org/

Fazendo backup e restaurando banco de dados do MySQL

Fazer backup de um banco de dados ao vivo é mais complicado. Você deve evitar copiar quaisquer dados de tabela enquanto o banco de dados estiver no meio de uma alteração.

Instruções sobre como fazer backup e restaurar um banco de dados do MySQL podem ser encontradas no Capítulo 12.

Segurança física

As ameaças à segurança que consideramos até agora se relacionam a coisas intangíveis como software, mas você não deve negligenciar a segurança física do sistema. Você precisa de ar-condicionado e proteção contra fogo, pessoas (desastradas ou criminosas), falta de energia e falha de rede.

Seu sistema deve estar bloqueado seguramente. Dependendo da escala de sua operação, isso poderia significar uma sala, uma jaula ou um armário. O pessoal que não precisar de acesso a essa sala de máquinas não deve ter acesso. Pessoas não-autorizadas poderiam, deliberada ou acidentalmente, desconectar cabos ou tentar pular mecanismos de segurança utilizando um disco de partida.

A água de sprinklers contra incêndio pode causar tanto dano aos dispositivos eletrônicos quanto o fogo. No passado, sistemas de supressão de fogo baseados em halon (um tipo de halocarboneto) eram utilizados para evitar esse problema. A produção de halon está agora proibida sob o Montreal Protocol on Substances That Deplete the Ozone Layer (Protocolo de Montreal sobre substâncias que esgotam a camada de ozônio), então, novos sistemas de supressão de fogo devem utilizar outras alternativas menos prejudiciais como argônio ou dióxido de carbono. Você pode ler mais sobre isso em http://epa.gov/Ozone/snap/fire/qa.html.

Faltas de energia breves e ocasionais são uma realidade na maioria dos lugares. Nas localidades com clima agreste e acima de fios-terra, ocorrem regularmente falhas longas. Se a operação contí-

nua de seus sistemas é importante, você deve investir em um no-break (Uninterruptible Power Supply – UPS). Um no-break que alimentará uma única máquina por 10 minutos custará menos de US$300,00. Para falhas mais longas ou mais equipamento, o gasto pode ser elevado. Faltas de energia prolongadas realmente exigem um gerador para também manter ligados os aparelhos de ar-condicionado além dos computadores.

Semelhantes às faltas de energia, as falhas de rede de minutos ou horas estão fora de seu controle e limitam-se a ocorrer ocasionalmente. Se sua rede for vital, faz sentido ter conexões com mais de um provedor de serviços da Internet. Custará mais ter duas conexões, mas isso deve significar que, em caso de falha, você apenas reduziu a capacidade em vez de tornar-se invisível.

Esses tipos de questões são algumas das razões por que você poderia querer considerar um esquema de co-localização para suas máquinas em uma instalação dedicada. Embora um negócio de tamanho médio talvez não seja capaz de justificar um UPS que garanta energia por mais de alguns minutos, diversas conexões de rede redundantes e sistemas de supressão de fogo, uma instalação de qualidade guardando máquinas de centenas de negócios semelhantes pode justificar.

A seguir

No Capítulo 16, veremos especificamente a autenticação – permitir que seus usuários provem sua identidade. Veremos alguns métodos diferentes, incluindo utilizar o PHP e o MySQL para autenticar seus visitantes.

16

Implementando autenticação com o PHP e o MySQL

ESTE CAPÍTULO DISCUTIRÁ COMO IMPLEMENTAR várias técnicas de PHP e MySQL para autenticar um usuário.

Os tópicos-chave deste capítulo incluem:

- Identificar visitantes;
- Implementar controle de acesso;
- Autenticação básica;
- Utilizar a autenticação básica no PHP;
- Utilizar a autenticação básica do `.htaccess` do Apache;
- Utilizar autenticação básica com o IIS;
- Utilizar autenticação `mod_auth_mysql`;
- Criar sua própria autenticação personalizada.

Identificando visitantes

A Web é uma mídia relativamente anônima, mas freqüentemente é útil saber quem está visitando seu site. Felizmente, para a privacidade dos visitantes, você pode descobrir muito pouco sobre eles sem o auxílio deles próprios. Com pouco trabalho, os servidores podem descobrir bastante sobre computadores e redes a que eles estão conectados. Um navegador Web normalmente se identificará informando ao servidor o navegador, a versão de navegador e o sistema operacional que você está executando. Usando JavaScript, você pode determinar a resolução e a profundidade de cor com que as telas dos visitantes estão configuradas e qual é o tamanho da janela do navegador Web.

Cada computador conectado à Internet tem um endereço IP único. A partir do endereço IP de um visitante, você talvez seja capaz de deduzir um pouco sobre ele. Você pode descobrir quem possui um IP e às vezes tem uma suposição razoável em relação à localização geográfica de um visitante. Alguns endereços serão mais úteis que outros. Geralmente, as pessoas com conexões permanentes com a Internet terão um endereço permanente. Os clientes que discam para um provedor de acesso da Internet (ISP) normalmente só obterão a utilização temporária de um dos endereços do ISP. Na próxima vez em que você vir esse endereço, talvez ele esteja sendo utilizado por um computador diferente, e a próxima vez que você vir esse visitante, provavelmente ele estará utilizando um endereço IP diferente. Os endereços IP não são tão úteis para identificar pessoas como eles parecem ser.

Felizmente para usuários da Web, nenhuma das informações que seus navegadores fornecem os identifica. Se você quiser saber o nome de um visitante ou outros detalhes, terá de perguntar a ele.

Muitos Web sites fornecem razões atraentes para fazer com que usuários forneçam suas informações pessoais. O jornal *The New York Times* (http://www.nytimes.com) fornece seu conteúdo gratuitamente, mas somente às pessoas dispostas a fornecer detalhes como nome, sexo e renda doméstica total. O site de discussão e noticiário para aficcionados de informática da Slashdot (http://www.slashdot.org) permite que usuários registrados participem de discussões sob um apelido e personalizem a interface que eles vêem. A maioria dos sites de comércio eletrônico registra as informações pessoais dos clientes quando eles fazem o primeiro pedido. Isso significa que não é necessário que um cliente as digite todas as vezes.

Tendo solicitado e recebido informações do visitante, você precisa de uma maneira de associar as informações ao mesmo usuário da próxima vez que ele visitar o site. Se estiver disposto a fazer a suposição de que somente uma pessoa visita o site a partir de uma conta particular em uma máquina particular e que cada visitante utiliza apenas uma máquina, você poderia armazenar um cookie na máquina do usuário para identificar o usuário. Isso certamente não é verdadeiro para todos os usuários – freqüentemente, muitas pessoas compartilham um computador e muitas pessoas utilizam mais de um. Pelo menos algumas das vezes, você precisará perguntar a um visitante quem ele é novamente. Além de perguntar a um usuário quem é ele, você também precisará pedir para ele fornecer algum nível de prova de que ele é quem afirma ser.

Como discutido no Capítulo 15, pedir a um usuário para provar sua identidade é chamado de *autenticação*. O método normal de autenticação utilizado nos Web sites é pedir aos visitantes que forneçam um nome único de login e uma senha. Normalmente, a autenticação é utilizada para ativar ou desativar acesso a determinadas páginas ou recursos, mas pode ser opcional ou utilizada para outros propósitos como personalização.

Implementando controle de acesso

O controle de acesso simples não é difícil de implementar. O código mostrado na Listagem 16.1 entrega uma das três possíveis saídas. Se o arquivo for carregado sem parâmetros, ele exibirá um formulário de HTML pedindo um nome de usuário e senha. Esse tipo de formulário é mostrado na Figura 16.1.

Figura 16.1 Nosso formulário HTML pede para os visitantes inserirem um nome de usuário e uma senha para o acesso.

Se os parâmetros estiverem presentes mas não corretos, uma mensagem de erro será exibida. Nossa mensagem de erro é mostrada na Figura 16.2.

Figura 16.2 Quando os usuários inserem detalhes incorretos, precisamos fornecer a eles uma mensagem de erro. Em um site real, talvez você deva fornecer uma mensagem um pouco mais amigável.

Se esses parâmetros estiverem presentes e corretos, a mensagem exibirá o conteúdo secreto. Nosso conteúdo de teste é mostrado na Figura 16.3.

Figura 16.3 Quando provido com detalhes corretos, nosso script exibirá o conteúdo.

O código para criar a funcionalidade mostrada na Figuras 16.1, 16.2 e 16.3 é mostrado na Listagem 16.1.

Listagem 16.1 **secret.php – PHP e HTML para fornecer um mecanismo de autenticação simples**

```
<?php
  //cria nomes abreviados para variáveis
@ $name = $HTTP_POST_VARS['name'];
@ $password = $HTTP_POST_VARS['password'];

  if(empty($name)||empty($password))
  {
    //O visitante precisa inserir um nome e uma senha
?>
    <h1>Please Log In</h1>
```

Listagem 16.1 **Continuação**

```
    This page is secret.
    <form method="post" action="secret.php">
    <table border="1">
    <tr>
      <th> Username <</th>
      <td> <input type="text" name="name"> </td>
    </tr>
    <tr>
      <th> Password </th>
      <td> <input type="password" name="password"> </td>
    </tr>
    <tr>
  .   <td colspan="2" align="center">
        <input type="submit" value="Log In">
      </td>
    </tr>
    </table>
   </form>
<?php
  }
  else if($name=='user'&&$password=='pass')
  {
    // a combinação nome do visitante e senha está correta
    echo '<h1>Here it is!</h1>';
    echo 'I bet you are glad you can see this secret page.';
  }
  else
  {
    // a combinação nome do visitante e senha está incorreta
    echo '<h1>Go Away!</h1>';
    echo 'You are not authorized to view this resource.';
  }
?>
```

O código de Listagem 16.1 fornecerá um mecanismo de autenticação simples para permitir que os usuários autorizados vejam uma página, mas ele tem alguns problemas significativos.

Esse script:

- Tem um nome de usuário e senha codificada no próprio script;
- Armazena a senha como texto simples;
- Protege somente uma página;
- Transmite a senha como texto simples.

Todas essas questões podem ser abordadas com graus variáveis de esforço e sucesso.

Armazenando senhas

Há muitos lugares melhores para armazenar nomes de usuário e senhas do que dentro do script. Dentro do script é difícil modificar os dados. É possível, mas também é uma má idéia, escrever um script para modificar a si próprio. Isso significaria ter um script no seu servidor, que é executado no seu servidor, mas é gravável ou modificável por outros. Armazenar os dados em outro arquivo no servidor permitirá escrever mais facilmente um programa para adicionar e remover usuários e alterar senhas.

Dentro de um script ou outro arquivo de dados, há um limite para o número de usuários que você pode ter sem afetar seriamente a velocidade do script. Se estiver considerando armazenar e

pesquisar por meio de um grande número de itens em um arquivo, em vez disso você deve considerar utilizar um banco de dados, como anteriormente discutido. De modo geral, se você quiser armazenar e pesquisar por meio de uma lista de mais de 100 itens, esses itens devem estar em um banco de dados em vez de em um arquivo simples.

Utilizar um banco de dados para armazenar nomes de usuário e senhas não faria o script muito mais complexo, mas permitiria autenticar muitos usuários diferentes rapidamente. Além disso, permitiria facilmente escrever um script para adicionar novos usuários, excluir usuários e permitir que usuários alterem senhas.

Um script para autenticar visitantes para uma página contra um banco de dados é dado na Listagem 16.2.

Listagem 16.2 **secretdb.php – Utilizamos MySQL para melhorar nosso mecanismo de autenticação simples**

```
<?php
  if(!isset($HTTP_POST_VARS['name'])&&!isset($HTTP_POST_VARS['password']))
  {
    //O visitante precisa inserir um nome e uma senha
?>
    <h1>Please Log In</h1>
    This page is secret.
    <form method="post" action="secretdb.php">
    <table border="1">
    <tr>
      <th> Username </th>
      <td> <input type="text" name="name"> </td>
    </tr>
    <tr>
      <th> Password </th>
      <td> <input type="password" name="password"> </td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" value="Log In">
      </td>
    </tr>
    </table>
    </form>
<?php
  }
  else
  {
    // conecta-se ao mySQL
    $mysql = mysql_connect( 'localhost', 'webauth', 'webauth' );
    if(!$mysql)
    {
      echo 'Cannot connect to database.';
      exit;
    }
    // seleciona o banco de dados apropriado
    $mysql = mysql_select_db( 'auth' );
    if(!$mysql)
    {
      echo 'Cannot select database.';
      exit;
    }

    // consulta o banco de dados para ver se há um registro correspondente
    $query = "select count(*) from auth where
```

Listagem 16.2 **Continuação**

```
            name = '$name' and
            pass = '$password'";

    $result = mysql_query( $query );
    if(!$result)
    {
      echo 'Cannot run query.';
      exit;
    }

    $count = mysql_result( $result, 0, 0 );

    if ( $count > 0 )
    {
     // a combinação nome do visitante e senha está correta
      echo '<h1>Here it is!</h1>';
      echo 'I bet you are glad you can see this secret page.';
    }
    else
    {
      // a combinação nome do visitante e senha está incorreta
      echo '<h1>Go Away!</h1>';
      echo 'You are not authorized to view this resource.';
    }
  }
?>
```

O banco de dados que estamos utilizando pode ser criado conectando-se ao MySQL como usuário root do MySQL e executando o conteúdo de Listagem 16.3.

Listagem 16.3 **`createauthdb.sql` – Essas consultas do MySQL criam o banco de dados auth, a tabela auth e dois usuários de exemplo**

```
create database auth;
use auth;
create table authorized_users ( name varchar(20),
                                password varchar(40),
                                        primary key     (name)
                              );
insert into authorized_users values ( 'username',
                                      'password' );

insert into authorized_users values ( 'testuser',
                                      sha1('password') );
grant select on auth.*
             to 'webauth'
             identified by 'webauth';
flush privileges;
```

Encriptando senhas

Independente de armazenarmos dados em um banco de dados ou em um arquivo, é um risco desnecessário armazenar as senhas como texto simples. Um algoritmo de hash de uma via pode fornecer um pouco mais de segurança com muito pouco esforço extra.

O PHP fornece diversas funções de hashing de uma via. A mais antiga e menos segura é o algoritmo Unix Crypt, fornecido pela função `crypt( )`. O algoritmo Message Digest 5 (MD5), imple-

mentado na função md5( ), é mais forte e está disponível na maior parte das versões do PHP. Se você não requer compatibilidade com versões mais antigas PHP, utilize o Secure Hash Algorithm 1 (SHA-1).

A função sha1( ) do PHP fornece uma função de hash criptográfica de uma via. O protótipo dessa função é:

```
string sha1 ( string str [, bool raw_output])
```

Com a string str, a função retornará uma string pseudo-aleatória de 40 caracteres. Se definir raw_output como true, você obterá uma string de dados binários com 20 caracteres. Por exemplo, a string "password", sha1( ) retorna "5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8". Essa string não pode ser criptografada e retornada para "password" nem por seu criador, então pode não parecer muito útil a princípio. O que torna sha1( ) útil é que a saída é determinística. Com a mesma string, sha1( ) retornará o mesmo resultado toda vez que for executada.

Em vez de ter código PHP do tipo

```
if( $name == 'username' &&
   $password == 'password' )
{
 //OK as senhas correspondem
}
```

é possível ter código do tipo

```
if( $name == 'username' &&
   sha1($password) == '5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8' ) )
{
 //OK as senhas correspondem
}
```

Não precisamos saber como a senha era antes de usar sha1( ). Precisamos saber apenas se a senha digitada é a mesma que aquela executada originalmente com sha1( ).

Como já mencionamos, é uma má idéia codificar manualmente nomes de usuário e senhas corretos em um script. Você deve utilizar um arquivo separado ou um banco de dados para armazená-los.

Se estiver utilizando um banco de dados do MySQL para armazenar a autenticação de dados, você pode usar a função do PHP sha1( ) ou a função do MySQL SHA1( ). O MySQL fornece uma maior variedade de algoritmos de hashing do que o PHP, mas eles não têm o mesmo propósito.

Para usar SHA1( ), reescreva a consulta SQL na Listagem 16.2 como:

```
select count(*) from authorized_users where
      name = '$username' and
      password = sha1('$password')
```

Essa consulta conta o número de linhas na tabela authorized_users que possuem o valor de nome igual ao conteúdo de $name e um valor de pass igual à saída fornecida por SHA1( ) aplicada ao conteúdo de $password. Assumindo que você fará as pessoas usarem nomes de usuários únicos, o resultado da consulta será 0 ou 1.

Lembre-se de que as funções de hash geralmente retornam dados de tamanho fixo. No caso de SHA1, são 40 caracteres, quando representada como string. Cerifique-se de que a coluna do seu banco de dados tenha essa largura.

Examinando novamente a Listagem 16.3, você pode ver que criamos um usuário ('username') com uma senha não-criptografada e outro usuário com uma criptografada ('testuser') para ilustrar as duas abordagens possíveis.

Protegendo várias páginas

Fazer um script que proteja mais de uma página é um pouco mais difícil. Como o HTTP é um protocolo sem informações de estado, não há nenhum vínculo ou associação automática entre solicitações subseqüentes provenientes da mesma pessoa. Isso dificulta mais o transporte de dados, como informações de autenticação que um usuário inseriu, de uma página para outra.

A maneira mais fácil de proteger várias páginas é utilizando mecanismos de controle de acesso fornecidos por seu servidor Web. Veremos esses mecanismos brevemente.

Para nós mesmos criarmos essa funcionalidade, poderíamos incluir partes do script mostrado na Listagem 16.1 em cada página que devemos proteger. Utilizando `auto_prepend_file` e `auto_append_file`, você pode automaticamente prefixar e acrescentar o código requerido a cada arquivo em diretórios particulares. O uso dessas diretivas foi discutido no Capítulo 5.

Se utilizarmos essa abordagem, o que acontece quando nossos visitantes vão para várias páginas dentro de nosso site? Não seria aceitável exigir que eles inserissem novamente seus nomes e senhas a cada página que desejassem visualizar.

Poderíamos acrescentar os detalhes que eles inseriram a cada hyperlink na página. Como nomes de usuários podem conter espaços ou outros caracteres que não são permitidos em URLs, você deve utilizar a função `urlencode( )` para codificar com segurança esses caracteres.

Mas ainda haveria alguns problemas com essa abordagem. Como os dados seriam incluídos nas páginas Web enviadas para o usuário e os URLs que eles visitam, as páginas protegidas que eles visitam serão visíveis a qualquer pessoa que utilize o mesmo computador e o recurso Voltar para ver páginas armazenadas em cache ou examinar a lista de histórico do navegador. Como estamos enviando a senha de um lado a outro para o navegador com cada página solicitada ou entregue, essas informações sigilosas estão sendo transmitidas com mais freqüência do que necessário.

Há duas boas maneiras de abordar esses problemas: autenticação básica de HTTP e sessões. A autenticação básica supera o problema de armazenamento em cache, mas o navegador ainda envia a senha para o navegador com cada solicitação. O controle de sessão supera esses dois problemas. Examinaremos a autenticação básica HTTP agora e o controle de sessão no Capítulo 22 e em mais detalhe no Capítulo 26.

Utilizando autenticação básica

Felizmente, como autenticar usuários é uma tarefa comum, há recursos de autenticação predefinidos no HTTP. Os scripts ou servidores Web podem solicitar autenticação de um navegador Web. O navegador Web é então responsável pela exibição de uma caixa de diálogo ou dispositivo semelhante para obter informações requeridas do usuário.

Embora o servidor Web solicite novos detalhes de autenticação a cada solicitação do usuário, o navegador Web não precisa solicitar os detalhes do usuário para cada página. O navegador geralmente armazena esses detalhes enquanto o usuário tiver uma janela de navegador aberta e automaticamente os reenvia para o servidor Web conforme necessário sem interação de usuário.

Esse recurso de HTTP é chamado *autenticação básica*. Você pode desencadear autenticação básica utilizando PHP ou utilizando mecanismos predefinidos do servidor Web. Veremos os métodos do PHP, do Apache e do IIS.

A autenticação básica transmite um nome de usuário e uma senha em texto simples, então ela não é muito segura. O HTTP 1.1 contém um método um pouco mais seguro conhecido como *autenticação de resenha (digest authentication)*, que utiliza um algoritmo de hashing (geralmente o MD5) para ocultar os detalhes da transação. A autenticação de resenha é suportada por muitos servidores Web e pela maioria dos navegadores Web atuais. Infelizmente, assim com vários recursos implementados mais recentemente, há vários navegadores mais antigos que ainda estão em uso e não suportam autenticação de resenha, e uma versão do padrão incluída no Microsoft IE e IIS que não é compatível com produtos não-Microsoft.

Além de ser pobremente suportada pelos navegadores Web instalados, a autenticação de resenha ainda não é muito segura. Tanto a autenticação básica como a de resenha fornecem um baixo nível de segurança. Nenhuma fornece qualquer garantia ao usuário de que ele está lidando com a máquina que pretendia acessar. As duas poderiam permitir que um cracker reproduzisse a mesma solicitação para o servidor. Como a autenticação básica transmite a senha do usuário como texto simples, ela permite que qualquer cracker capaz de capturar pacotes personifique o usuário para fazer qualquer solicitação.

A autenticação básica fornece um (baixo) nível de segurança semelhante àquele comumente utilizado para conectar-se a máquinas via telnet ou ftp, transmitindo senhas em texto simples. A autenticação de resenha é uma pouco mais segura, encriptando senhas antes de transmiti-las.

Quando combinamos autenticação básica com SSL e certificados digitais, todas as partes de uma transação pela Web podem ser protegidas por segurança forte. Se quiser segurança forte, você deve ler o próximo capítulo. Entretanto, para muitas situações, um método rápido, mas relativamente inseguro, como a autenticação básica é apropriado.

A autenticação básica protege um reino identificado e requer que os usuários forneçam um nome de usuário e uma senha válidos. Os reinos são identificados de modo que mais de um reino possa estar no mesmo servidor. Diferentes arquivos ou diretórios no mesmo servidor podem ser parte de reinos diferentes, cada um protegido por um conjunto diferente de nomes e senhas. Reinos identificados também permitem agrupar diversos diretórios em um host ou host virtual como um reino e protegem todos eles com uma senha.

Utilizando autenticação básica no PHP

Os scripts de PHP são geralmente compatíveis com várias plataformas, mas utilizar a autenticação básica conta com um conjunto de variáveis de ambiente pelo servidor. A fim de que um script de autenticação HTTP execute no Apache utilizando o PHP como um Apache Module ou no IIS utilizando o PHP como um módulo ISAPI, ele precisa detectar o tipo de servidor e comportar-se de maneira ligeiramente diferente. O script na Listagem 16.4 será executado nos dois servidores,

Listagem 16.4 **http.php** – PHP pode desencadear autenticação básica de HTTP

```
<?php

// se estivermos utilizando IIS, precisamos configurar $PHP_AUTH_USER e $PHP_AUTH_PW
if (substr($SERVER_SOFTWARE, 0, 9) == 'Microsoft' &&
    !isset($PHP_AUTH_USER) &&
    !isset($PHP_AUTH_PW) &&
    substr($HTTP_AUTHORIZATION, 0, 6) == 'Basic '
   )
{
  list($PHP_AUTH_USER, $PHP_AUTH_PW) =
    explode(':', base64_decode(substr($HTTP_AUTHORIZATION, 6)));
}

// Substitua essa instrução IF por uma consulta a banco de dados ou algo semelhante
if ($PHP_AUTH_USER != 'user' || $PHP_AUTH_PW != 'pass')
{
  // o visitante ainda não forneceu detalhes, ou sua
  // combinação de nome e senha não está correta

  header('WWW-Authenticate: Basic realm="Realm-Name"');
  if (substr($SERVER_SOFTWARE, 0, 9) == 'Microsoft')
    header('Status: 401 Unauthorized');
  else
    header('HTTP/1.0 401 Unauthorized');
```

Listagem 16.4 **Continuação**

```
    echo '<h1>Go Away!</h1>';
    echo 'You are not authorized to view this resource.';
}
else
{
    // o visitante forneceu os detalhes corretos
    echo '<h1>Here it is!</h1>';
    echo '<p>I bet you are glad you can see this secret page.</p>';
}
?>
```

O código na Listagem 16.4 atua de uma maneira muito semelhante às listagens anteriores neste capítulo. Se o usuário ainda não forneceu as informações de autenticação, essas informações serão solicitadas. Se o usuário tiver fornecido informações incorretas, ele recebe uma mensagem de rejeição. Se ele fornecer um par nome-senha correspondente, o conteúdo da página é apresentado.

O usuário verá uma interface um pouco diferente das listagens anteriores. Não estamos fornecendo um formulário HTML para as informações de login. O navegador do usuário apresentará ao usuário uma caixa de diálogo. Algumas pessoas vêem isso como uma melhora; outros prefeririam ter controle completo sobre os aspectos visuais da interface. A caixa de diálogo de login que o Internet Explorer fornece é mostrada na Figura 16.4.

Como a autenticação está sendo auxiliada por recursos integrados ao navegador, os navegadores escolheram exercitar alguma discrição na maneira como as tentativas de autorização malsucedidas são tratadas. O Internet Explorer permite ao usuário tentar autenticar três vezes antes de exibir a página de rejeição. O Netscape permitirá ao usuário tentar um número ilimitado de vezes, abrindo uma caixa de diálogo perguntando "Authorization failed. Retry?" entre as tentativas. O Netscape só exibe a página de rejeição se o usuário clicar em Cancel.

Assim como o código das Listagens 16.1 e 16.2, poderíamos incluir esse código em páginas que quiséssemos proteger, ou automaticamente prefixá-lo para cada arquivo em um diretório.

Figura 16.4 O navegador do usuário é responsável pela aparência da caixa de diálogo ao utilizar autenticação de HTTP.

Utilizando autenticação básica com arquivos .htaccess do Apache

Podemos alcançar resultados muito semelhantes aos do script anterior sem escrever um script de PHP.

O servidor Web Apache contém vários módulos de autenticação diferentes que podem ser utilizados para decidir a validade dos dados inseridos por um usuário. O mais fácil de utilizar é mod_auth, que compara pares de nome-senha a linhas em um arquivo de texto no servidor.

Para obter a mesma saída que o script anterior, precisamos criar dois arquivos HTML separados, um para o conteúdo e um para a página de rejeição. Pulamos alguns elementos de HTML nos exemplos anteriores, mas na realidade deveríamos incluir as tags <html> e <body> quando estamos gerando HTML.

A Listagem 16.5 mostra o conteúdo que usuários autorizados vêem. Chamamos esse arquivo de content.html. A Listagem 16.6 contém a página de rejeição. Chamamos essa página de rejection.html. Mostrar uma página em caso de erros é opcional, mas é um toque gentil e profissional se você colocar algo útil nela. Dado que essa página será mostrada quando um usuário tentar entrar em uma área protegida mas for rejeitado, o conteúdo útil poderia incluir instruções sobre como registrar uma senha ou como criar uma senha nova ou receber a senha correta pelo e-mail se ela foi esquecida.

Listagem 16.5 **content.html – Nosso conteúdo de exemplo**

```
<html><body>
<h1>Here it is!</h1>
<p>I bet you are glad you can see this secret page.</p>
</body></html>
```

Listagem 16.6 **rejection.html – Nosso exemplo de página de erro 401**

```
<html><body>
<h1>Go Away!</h1>
<p>You are not authorized to view this resource.</p>
</body></html>
```

Não há nada de novo nesses arquivos. O arquivo que interessa neste exemplo é a Listagem 16.7. Esse arquivo deve se chamar .htaccess e controlará o acesso aos arquivos e quaisquer subdiretórios no seu diretório.

Listagem 16.7 **htaccess – Um arquivo .htaccess pode determinar muitas definições de configuração do Apache, incluindo a ativação de autenticação**

```
ErrorDocument 401 /chapter14/rejection.html
AuthUserFile /home/book/.htpass
AuthGroupFile /dev/null
AuthName "Realm-Name"
AuthType Basic
require valid-user
```

A Listagem 16.7 é um arquivo .htaccess para ativar autenticação básica em um diretório. Muitas configurações podem ser feitas em um arquivo .htaccess, mas todas as seis linhas no nosso exemplo se relacionam à autenticação.

A primeira linha

```
ErrorDocument 401 /chapter14/rejection.html
```

diz ao Apache que documento exibir para os visitantes que não conseguem se autenticar. Você pode utilizar outras diretivas de ErrorDocument para fornecer suas próprias páginas para outros erros de HTTP como o 404. A sintaxe é:

```
ErrorDocument URL_do_número_do_erro
```

Para uma página tratar o erro 401, é importante que o URL fornecido esteja publicamente disponível. Não seria muito útil fornecer uma página personalizada de erro para informar às pessoas que sua autorização falhou se a página estiver bloqueada em um diretório em que elas precisam autenticar com sucesso para ver.

A linha

```
AuthUserFile /home/book/.htpass
```

diz ao Apache onde localizar o arquivo que contém senhas dos usuários autorizados. Esse arquivo é freqüentemente chamado `.htpass`, mas você pode dar a ele o nome que preferir. O importante não é o nome desse arquivo, mas onde ele está armazenado. Esse arquivo não deve ser armazenado dentro da árvore Web – em algum lugar que as pessoas possam fazer download dele via servidor Web. Nosso arquivo `.htpass` de exemplo é mostrado na Listagem 16.8.

Bem como especificar usuários individuais que são autorizados, é possível especificar que somente usuários autorizados que se classificam dentro de grupos específicos podem acessar recursos. Escolhemos não especificar, então a linha

```
AuthGroupFile /dev/null
```

configura o `AuthGroupFile` para apontar para `/dev/null`, um arquivo especial nos sistemas Unix que seguramente é nulo.

Como no exemplo de PHP, para utilizar autenticação HTTP, precisamos nomear nosso reino da seguinte maneira:

```
AuthName "Realm-Name"
```

Você pode escolher o nome de reino que preferir, mas lembre-se de que o nome será mostrado para seus visitantes. Para tornar óbvio que o nome no exemplo deve ser alterado, o nosso se chama `"Realm-Name"`.

Como vários métodos de autenticação diferentes são suportados, precisamos especificar qual método de autenticação estamos utilizando.

Estamos utilizando a autenticação Basic como especificado por esta diretiva:

```
AuthType Basic
```

Precisamos especificar quem tem permissão de acesso. Poderíamos especificar usuários particulares, grupos particulares ou, como fizemos, simplesmente permitir qualquer acesso de usuário autenticado.

A linha

```
require valid-user
```

especifica que qualquer usuário válido deve ter acesso permitido.

Listagem 16.8 **htpass – O arquivo Password armazena nomes de usuário e a senha encriptada de cada usuário**

```
user1:0nRp9M80GS7zM
user2:nC13sOTOhp.ow
user3:yjQMCPWjXFTzU
user4:LOmlMEi/hAme2
```

Cada linha no arquivo `.htpass` contém um nome de usuário, um dois-pontos e a senha encriptada desse usuário.

O conteúdo exato do arquivo `.htpass` variará. Para criá-lo, você utiliza um programa pequeno chamado `htpasswd` que insere a distribuição Apache.

O programa htpasswd é utilizado de uma das seguintes maneiras:

```
htpasswd [-cmdps] arquivodesenha nomedousuário
```

ou:

```
htpasswd -b[cmdps] arquivodesenha nomedousuário
```

A única opção que você precisa utilizar é -c. Utilizar -c instrui htpasswd a criar o arquivo. Você deve utilizar isso para o primeiro usuário que adicionar. Tenha o cuidado de não utilizá-lo para outros usuários porque se o arquivo existir, htpasswd o excluirá e criará um novo.

As opções m, d, p ou s podem ser utilizadas se você quiser especificar qual algoritmo de criptografia (incluindo nenhuma criptografia) gostaria de utilizar.

A opção b instrui o programa a esperar a senha como um parâmetro, em vez de enviar um prompt para ele. Isso é útil se você quiser chamar htpasswd não-interativamente como parte de um processo de lote, mas não deve ser utilizado se você estiver chamando htpasswd a partir da linha de comando.

Os seguintes comandos criaram o arquivo mostrado na Listagem 16.8:

```
htpasswd -bc /home/book/.htpass user1 pass1
htpasswd -b /home/book/.htpass user2 pass2
htpasswd -b /home/book/.htpass user4 pass3
htpasswd -b /home/book/.htpass user4 pass4
```

Note que htpasswd pode não estar no seu caminho. Caso não esteja, você deve fornecer o caminho completo. Em geral ele pode ser encontrado em /usr/local/apache/bin.

Esse tipo de autenticação é fácil de configurar, mas há alguns problemas em utilizar um arquivo .htaccess dessa maneira.

Os usuários e as senhas são armazenados em um arquivo de texto. Toda vez que um navegador solicitar um arquivo protegido pelo arquivo .htaccess, o servidor deve analisar sintaticamente o arquivo .htaccess e, então, analisar sintaticamente o arquivo de senha, tentando fazer correspondência com nome de usuário e senha. Em vez de utilizar um arquivo .htaccess, poderíamos fazer as mesmas especificações em nosso arquivo httpd.conf – o principal arquivo de configuração para o servidor Web. Um arquivo .htaccess é analisado sintaticamente toda vez que um arquivo é solicitado. O arquivo httpd.conf só é analisado sintaticamente quando o servidor é inicializado pela primeira vez. Isso será mais rápido, mas significa que se quisermos fazer alterações, precisamos parar e reiniciar o servidor.

Independente de onde armazenamos as diretivas de servidor, o arquivo de senha ainda precisa ser pesquisado por cada solicitação. Isso significa que, como outras técnicas que examinamos que utilizam um arquivo simples, essa não seria apropriada para centenas ou milhares de usuários.

Utilizando autenticação básica com IIS

Como o Apache, o IIS suporta autenticação de HTTP. O Apache utiliza a abordagem de UNIX e é controlado editando arquivos de texto e, como você poderia esperar, selecionar opções nas caixas de diálogo controla a configuração de IIS.

Utilizando o Windows XP, você altera a configuração do Internet Information Server 5 (IIS5) utilizando o Internet Services Manager. Você pode localizar esse utilitário escolhendo Administrative Tools no Control Panel.

O Internet Services Manager parece com a janela mostrada na Figura 16.5. O controle de árvore no lado esquerdo mostra que na máquina chamada windows-server, estamos executando vários serviços. O site em que estamos interessados é o Web site padrão. Dentro desse Web site, temos um diretório chamado protected. Dentro desse diretório está um arquivo chamado content.html.

Figura 16.5 O Microsoft Management Console permite configurar o Internet Information Server 5.

Para adicionar autenticação básica ao diretório `protected`, clique com o botão direito do mouse nele e selecione Properties no menu de contexto.

A caixa de diálogo Properties permite alterar muitas configurações desse diretório. As duas guias em que estamos interessados são Directory Security e Custom Errors. Uma das opções na guia Directory Security é Anonymous Access e Authentication Control. Pressionar esse botão Edit abrirá a caixa de diálogo mostrada na Figura 16.6.

Dentro dessa caixa de diálogo, podemos desativar o acesso anônimo e ativar autenticação básica. Com as configurações mostradas na Figura 16.6, somente as pessoas que fornecem uma senha e um nome apropriados podem visualizar arquivos nesse diretório.

A fim de duplicar o comportamento dos exemplos anteriores, também forneceremos uma página para informar os usuários de que os detalhes da sua autenticação não estavam corretos. Fechar a caixa de diálogo Authentication Methods permitirá escolher a guia Custom Errors.

Figura 16.6 O IIS5 permite acesso anônimo por padrão, mas permite ativar autenticação.

A guia Custom Errors, mostrada na Figura 16.7, associa erros a mensagens de erro. Aqui, armazenamos o mesmo arquivo de rejeição que utilizamos anteriormente, `rejection.html`, mostrado na Listagem 16.6. O IIS é capaz de fornecer uma mensagem de erro mais específica do que o Apache, mostrando o código do erro de HTTP que ocorreu e uma razão pela qual ele ocorreu. Para o erro 401, que representa autenticação falha, o IIS fornece cinco razões diferentes. Poderíamos fornecer

mensagens diferentes para cada uma, mas escolhemos só substituir as duas que vão ocorrer neste exemplo com nossa página de rejeição.

Isso é tudo que precisamos fazer para requerer autenticação para esse diretório utilizando o IIS5. Como inúmeros outros programas Windows, isso é mais fácil de configurar do que o software UNIX semelhante, mas mais difícil de copiar de máquina para máquina ou de diretório para diretório. Além disso, é fácil configurá-lo acidentalmente de uma maneira que torne a máquina insegura.

O principal defeito com a abordagem do IIS é que ele autentica usuários Web comparando seus detalhes de login com contas da máquina. Se quisermos permitir que um usuário "john" efetue logon com a senha "password", precisamos criar uma conta de usuário na máquina ou em um domínio, com esse nome e senha. Você precisa ser muito cuidadoso ao criar contas para a autenticação Web de modo que os usuários só tenham os direitos de conta que eles precisam para visualizar páginas Web e não tenham outros direitos, como acesso de telnet.

Figura 16.7 A guia Custom Errors permite associar páginas personalizadas de erro a eventos de erro.

Utilizando autenticação `mod_auth_mysql`

Como já mencionado, utilizar `mod_auth` com Apache é fácil de configurar e eficiente. Como os nomes de usuário são armazenados em um arquivo de texto, isso não é realmente prático para sites ocupados com vários usuários.

Felizmente, você pode ter a maior parte da facilidade do `mod_auth` e a velocidade de um banco de dados utilizando `mod_auth_mysql`. Esse módulo funciona da mesma maneira que `mod_auth`, mas como utiliza um banco de dados do MySQL em vez de um arquivo de texto, ele pode pesquisar grandes listas de usuários rapidamente.

Para utilizá-lo, você precisará compilar e instalar o módulo no seu sistema ou pedir a seu administrador de sistema para instalá-lo.

Instalando `mod_auth_mysql`

Para utilizar `mod_auth_mysql`, você precisará configurar o MySQL e o Apache de acordo com as instruções no Apêndice A, mas adicione alguns passos extras. Há instruções muito boas nos arquivos `README` e `USAGE` que estão na distribuição, mas eles se referem a versões anteriores em alguns lugares. Eis um resumo.

1. Obtenha o repositório de arquivo de distribuição para o módulo. Ele está no CD-ROM que veio com o livro, mas você pode sempre obter a última versão em http://sourceforge.net/projects/mod-auth-mysql/.
2. Descompacte o código-fonte com unzip e untar.
3. Mude o diretório `mod_auth_mysql`, execute `make` e então `make install`. Você pode precisar mudar os locais de instalação para o MySQL no arquivo make (`MakeFile`).
4. Acrescente esta linha a `httpd.conf`:

   ```
   LoadModule mysql_auth_module libexec/mod_auth_mysql.so
   ```

   para carregar o módulo dinamicamente no Apache.
5. Crie um banco de dados e um tabela no MySQL para conter as informações de autenticação. Não precisa ser um banco de dados ou tabela separados; você pode utilizar uma tabela existente como o banco de dados `auth` do exemplo anterior neste capítulo.
6. Acrescente uma linha ao seu arquivo `httpd.conf` para fornecer a `mod_auth_mysql` os parâmetros necessários para conectar ao MySQL. A diretiva se parecerá com:

   ```
   Auth_MySQL_Info hostname user password
   ```

A compilação funcionou?

A maneira mais fácil de verificar se a compilação funcionou é ver se o Apache inicia. Para iniciar o Apache, se você tiver suporte SSL, digite:

```
/usr/local/apache/bin/apachectl startssl
```

Se não tiver suporte SSL, você pode digitar:

```
/usr/local/apache/bin/apachectl start
```

Se ele iniciar com a diretiva `Auth_MySQL_Info` no arquivo `httpd.conf`, `mod_auth_mySQL` foi adicionado com sucesso.

Utilizando `mod_auth_mysql`

Depois de instalar o módulo com sucesso, utilizá-lo não é mais difícil do que utilizar `mod_auth`. A Listagem 16.9 mostra um arquivo `.htaccess` de exemplo que autenticará os usuários com senhas encriptadas armazenadas no banco de dados criado anteriormente neste capítulo.

Listagem 16.9 **htaccess – Esse arquivo .htaccess autentica usuários com base em um banco de dados do MySQL**

```
ErrorDocument 401 /chapter14/rejection.html

AuthName "Realm Name"
AuthType Basic

Auth_MySQL_DB auth
Auth_MySQL_Encryption_Types MySQL
Auth_MySQL_Password_Table auth
Auth_MySQL_Username_Field name
Auth_MySQL_Password_Field pass

require valid-user
```

Você pode ver que grande parte da Listagem 16.9 é a mesma que a Listagem 16.7. Ainda estamos especificando um documento de erro para exibir no caso de erro 401 (quando a autenticação falha). Novamente especificamos a autenticação básica e fornecemos um nome ao reino. Como na Listagem 16.7, permitiremos qualquer acesso de usuário válido e autenticado.

Como estamos utilizando `mod_auth_mysql` e não queríamos utilizar todas as configurações padrão, temos algumas diretivas para especificar como isso deve funcionar. `Auth_MySQL_DB`, `Auth_MySQL_Password_Table`, `Auth_MySQL_Username_Field` e `Auth_MySQL_Password_Field` especificam o nome do banco de dados, a tabela, o campo de nome de usuário e campos de senha, respectivamente.

Incluímos a diretiva `Auth_MySQL_Encryption_Types` para especificar que queremos utilizar criptografia de senha do MySQL. Valores aceitáveis são `Plaintext`, `Crypt_DES` ou `MySQL`. `Crypt_DES` é o padrão e utiliza senhas criptografadas UNIX DES.

Da perspectiva de usuário, esse exemplo `mod_auth_mysql` funcionará exatamente da mesma maneira que o exemplo `mod_auth`. O navegador Web apresentará ao usuário uma caixa de diálogo. Se o usuário se autenticar com sucesso, o conteúdo será mostrado. Se falhar, ele receberá a página de erro.

Para muitos Web sites, `mod_auth_mysql` é ideal. Ele é rápido, relativamente fácil de implementar e permite utilizar qualquer mecanismo conveniente para adicionar entradas de banco de dados para novos usuários. Para obter mais flexibilidade e a capacidade de aplicar controle refinado a partes de páginas, talvez você queira implementar sua própria autenticação utilizando o PHP e o MySQL.

Criando sua própria autenticação personalizada

Neste capítulo, examinamos a criação de nossos próprios métodos de autenticação incluindo algumas falhas e compromissos, e a utilização de métodos de autenticação predefinidos, que são menos flexíveis do que escrever seu próprio código. Mais adiante no livro, quando abrangermos o controle de sessão, você será capaz de escrever sua própria autenticação personalizada com menos compromissos que neste capítulo.

No Capítulo 22, desenvolveremos um sistema de autenticação de usuário simples que evita alguns problemas com que nos deparamos aqui utilizando as sessões para monitorar variáveis entre páginas.

No Capítulo 26, aplicamos essa abordagem a um projeto do mundo real e examinamos como ela pode ser utilizada para implementar um sistema de autenticação refinada.

Leitura adicional

Os detalhes da autenticação de HTTP são especificados por RFC 2617, que está disponível em:

http://www.rfc-editor.org/rfc/rfc2617.txt

A documentação para `mod_auth`, que controla a autenticação básica no Apache, pode ser encontrada em:

http://www.apache.org/docs/mod/mod_auth.html

A documentação para o `mod_auth_mysql` está dentro do repositório de arquivos de download. O arquivo de download é tão pequeno que vale a pena descarregá-lo, mesmo que você queira apenas descobrir mais algumas informações.

A seguir

O próximo capítulo explica como salvaguardar dados em todas as etapas do processamento dos dados, desde a entrada, passando pela transmissão até o armazenamento. Esse capítulo inclui o uso de SSL, certificados digitais e criptografia.

17

Implementando transações seguras com PHP e MySQL

NESTE CAPÍTULO, EXPLICAREMOS COMO LIDAR seguramente com dados de usuário a partir da entrada, ao longo da transmissão e no armazenamento. Isso nos permitirá implementar seguramente uma transação de ponta a ponta entre nós e um usuário.

Os tópicos-chave deste capítulo incluem:

- Fornecendo transações seguras;
- Utilizando Secure Sockets Layer (SSL);
- Fornecendo armazenamento seguro;
- Determinando por que você está armazenando números de cartão de crédito;
- Utilizando criptografia no PHP.

Fornecendo transações seguras

Fornecer transações seguras utilizando a Internet é uma questão de examinar o fluxo das informações no sistema e assegurar que em cada ponto suas informações estejam seguras. No contexto de segurança de rede, não há nada absoluto. Nenhum sistema jamais será impenetrável. Por seguro, queremos dizer que o nível de esforço requerido para comprometer um sistema ou transmissão é alto comparado ao valor das informações envolvidas.

Se devemos orientar nossos esforços de segurança efetivamente, precisamos examinar o fluxo das informações por todas as partes do sistema. O fluxo das informações de usuário em uma aplicação típica, escrita utilizando o PHP e o MySQL, é mostrado na Figura 17.1.

Os detalhes de cada transação ocorrendo no sistema variarão, dependendo do projeto do sistema, dos dados do usuário e das ações que desencadearam a transação. Você pode examinar todos esses de uma maneira semelhante. Cada transação entre uma aplicação Web e um usuário começa com o navegador do usuário que envia uma solicitação pela Internet para o servidor Web. Se a página for um script de PHP, o servidor Web delegará o processamento da página para o mecanismo de PHP.

O script de PHP pode ler ou gravar dados para disco. Ele também pode incluir (`include( )`) ou requerer (`require( )`) outros arquivos de PHP ou de HTML. Esse script também enviará consultas de SQL para o daemon do MySQL e receberá respostas. O mecanismo do MySQL é responsável pela leitura e gravação de seus próprios dados no disco.

Figura 17.1 As informações do usuário são armazenadas ou processadas pelos seguintes elementos do ambiente de uma típica aplicação Web.

Esse sistema tem três partes principais:

- A máquina do usuário;
- A Internet;
- Seu sistema.

Veremos considerações de segurança para cada parte separadamente, mas é óbvio que a máquina do usuário e a Internet estão amplamente fora de seu controle.

A máquina do usuário

Do nosso ponto de vista, a máquina do usuário está executando um navegador Web. Não temos nenhum controle sobre outros fatores como a segurança com que a máquina está configurada. Precisamos ter em mente que a máquina talvez seja muito insegura ou até mesmo um terminal compartilhado em uma biblioteca, escola ou café.

Muitos navegadores diferentes estão disponíveis, cada um com capacidades ligeiramente diferentes. Se considerarmos somente as versões recentes dos dois navegadores mais populares, a maioria das diferenças entre eles só afeta a maneira como a HTML será gerada e exibida, mas há questões de segurança ou funcionalidade que precisamos considerar.

Você deve notar que algumas pessoas desativarão recursos que consideram um risco de segurança ou de privacidade, como Java, cookies ou JavaScript. Se utilizar esses recursos, você deve testar se sua aplicação resolve de forma elegante a maneira como se apresenta para pessoas sem esses recursos, ou considerar fornecer uma interface menos rica em recursos que permita que essas pessoas utilizem seu site.

Os usuários fora dos Estados Unidos e Canadá talvez tenham navegadores Web que suportem somente criptografia de 40 bits. Embora o governo norte-americano tenha mudado a lei em janeiro de 2000 para permitir a exportação de criptografia forte (para países sem embargo econômico) e versões de 128 bits estejam agora disponíveis para a maioria dos usuários, algumas delas não terão a atualização. A menos que você esteja oferecendo garantias de segurança para usuários no texto do site, essa necessidade não se aplica a você como um desenvolvedor Web. O SSL automaticamente negociará a permissão para que o seu servidor e o navegador do usuário comuniquem-se no nível mais seguro que ambos entendam.

Não podemos estar certos de que estamos lidando com um navegador Web que se conecta ao nosso site por meio da interface pretendida. As solicitações para o site talvez venham de outro site roubando imagens ou conteúdo ou de uma pessoa utilizando software como cURL para pular medidas de segurança.

Veremos a biblioteca cURL, que pode ser utilizada para simular conexões a partir de um navegador, no Capítulo 19. Isso é útil para nós como desenvolvedores, mas também pode ser utilizado maliciosamente.

Embora não possamos alterar nem controlar a maneira como as máquinas dos usuários estão configuradas, precisamos ter isso em mente. A variabilidade de máquinas de usuário talvez seja um fator ao decidir quanta funcionalidade fornecemos via script do lado servidor (como PHP) e quanta funcionalidade fornecemos via script do lado cliente (como JavaScript).

A funcionalidade fornecida pelo PHP pode ser compatível com o navegador de cada usuário, já que o resultado final é meramente uma página HTML. Utilizar qualquer coisa além de um código JavaScript muito básico envolverá levar em conta as diferentes capacidades de versões individuais do navegador.

Da perspectiva da segurança, seria melhor utilizar script do lado servidor para itens como validação de dados porque, dessa maneira, nosso código-fonte não será visível ao usuário. Se validarmos os dados em JavaScript, os usuários serão capazes de ver o código e talvez se apropriar dele indevidamente.

Os dados que precisam ser guardados podem ser armazenados em nossas próprias máquinas, como arquivos ou registros de banco de dados, ou nas máquinas dos nossos usuários, como cookies. Examinaremos a utilização de cookies para armazenar alguns dados limitados (uma chave de sessão) no Capítulo 22.

A maior parte dos dados que armazenamos deve residir no servidor Web ou em nosso banco de dados. Há várias boas razões para armazenar a menor quantidade possível de informações na máquina do usuário. Se as informações estiverem fora do sistema, você não tem nenhum controle sobre a segurança com que elas estão armazenadas, você não pode se certificar de que o usuário não as excluirá e não pode impedir que o usuário as modifique em uma tentativa de confundir o sistema.

A Internet

Da mesma maneira que acontece com a máquina do usuário, você tem muito pouco controle sobre as características da Internet, mas isso não significa que você pode ignorar essas características ao projetar o sistema.

A Internet tem muitos bons recursos, mas é uma rede inerentemente insegura. Ao enviar informações de um ponto a outro, você precisa ter em mente que outros poderiam visualizar ou alterar informações que você está transmitindo, como discutimos no Capítulo 15. Com isso em mente, você pode decidir que ação tomar.

Sua resposta poderia ser:

- Transmitir as informações de qualquer jeito, sabendo que talvez elas não sejam privadas e talvez cheguem alteradas.
- Assinar digitalmente as informações antes de transmiti-las, a fim de protegê-las de falsificação.
- Encriptar ou assinar as informações antes de transmiti-las para mantê-las privadas ou protegê-las contra adulteração.
- Decidir que suas informações são sigilosas a ponto de não permitir qualquer chance de interceptação e encontrar outro caminho de distribuir suas informações.

A Internet também é uma mídia relativamente anônima. É difícil ter certeza de que a pessoa com quem você está lidando é quem ela afirma ser. Mesmo que você possa garantir a identidade de um usuário para sua própria satisfação, talvez seja difícil provar isso em um fórum como um tribunal. Isso causa problemas como o repúdio, que discutimos no Capítulo 15.

Em resumo, privacidade e repúdio são grandes questões ao conduzir transações pela Internet.

Há pelo menos duas maneiras diferentes de armazenar as informações que fluem para e a partir do servidor Web pela Internet:

- SSL (Secure Sockets Layer)
- S-HTTP (Secure Hypertext Transfer Protocol)

Essas duas tecnologias oferecem mensagens e autenticação privadas resistentes à adulteração, mas o SSL está prontamente disponível e é bastante utilizado enquanto o S-HTTP realmente não decolou. Veremos o SSL em detalhes mais adiante neste capítulo.

Seu sistema

Seu sistema é a parte do universo que você controla. Ele é representado pelos componentes dentro da caixa retangular, como foi mostrado anteriormente na Figura 17.1. Esses componentes poderiam estar fisicamente separados em uma rede ou existir todos juntos em uma mesma máquina física.

É relativamente seguro não se preocupar com a segurança das informações enquanto os vários produtos de terceiros que utilizamos para entregar nosso conteúdo Web as estão tratando. Os autores desses programas provavelmente pensaram mais tempo do que você possa imaginar sobre os aspectos particulares do processo tratados pelo produto. Contanto que esteja utilizando uma versão atualizada de um produto bem conhecido, você poderá encontrar qualquer problema bem conhecido utilizando o Google ou seu mecanismo de pesquisa Web favorito. Manter-se atualizado com essas informações deve ser sua prioridade.

Se a instalação e a configuração são partes da sua função, você precisa preocupar-se com a maneira como o software é instalado e configurado. Muitos equívocos feitos na segurança são resultado de não seguir os avisos na documentação ou envolvem questões gerais de administração do sistema, assunto para um outro livro. Adquira um bom livro sobre administração do sistema operacional que você pretende utilizar ou empregue um especialista de administração de sistema.

Algo a ser considerado ao instalar o PHP é que geralmente é mais seguro, e muito mais eficiente, instalar o PHP como um módulo SAPI em seu servidor Web do que executá-lo via interface CGI.

Sua principal preocupação deve ser o que seus próprios scripts fazem ou não fazem. Que dados potencialmente sigilosos nossa aplicação transmite para o usuário pela Internet? Que dados sigilosos pedimos aos usuários para transmitir? Se estamos transmitindo informações que devem ser uma transação privada entre nós e nossos usuários ou que devem ser difíceis para um intermediário modificar, devemos considerar utilizar o SSL.

Já falamos sobre como utilizar SSL entre o computador do usuário e o servidor. Você também deve pensar na situação em que está transmitindo dados de um componente do sistema para outro em uma rede. Um exemplo típico surge quando o banco de dados do MySQL reside em uma máquina diferente do servidor Web. O PHP conecta-se ao servidor do MySQL via TCP/IP e essa conexão será não-criptografada. Se essas duas máquinas estiverem em uma rede local privada, você precisa assegurar que a rede é segura. Se as máquinas estiverem comunicando-se via Internet, seu sistema provavelmente ficará lento e você precisará tratar essa conexão da mesma maneira que outras conexões pela Internet.

É importante que quando nossos usuários pensarem estar se comunicando conosco, eles realmente estejam. Registrar um certificado digital protegerá nossos visitantes contra *spoofing* (alguém simulando nosso site), permitirá utilizar o SSL sem que os usuários vejam uma mensagem de advertência e fornecerá um ar de respeitabilidade ao nosso empreendimento on-line.

Nossos scripts verificam cuidadosamente os dados inseridos pelos usuários? Temos cuidado em armazenar as informações seguramente? Responderemos a essas perguntas nas próximas seções deste capítulo.

Utilizando Secure Sockets Layer (SSL)

O conjunto de protocolos de Secure Sockets Layer foi originalmente projetado pela Netscape para facilitar comunicação segura entre servidores e navegadores Web. Ele tem sido desde então adotado como o método padrão não-oficial para navegadores e servidores trocarem informações sigilosas.

Tanto a versão 2 como a 3 do SSL são bem suportadas. A maioria dos servidores Web inclui funcionalidade de SSL ou pode aceitá-la como um módulo suplementar. O Internet Explorer e o Netscape suportam o SSL desde a versão 3.

Os protocolos de rede e o software que os implementa normalmente são organizados como uma pilha de camadas. Cada camada pode passar dados para a camada acima ou abaixo e solicitar serviços da camada acima ou abaixo. A Figura 17.2 mostra essa pilha de protocolos.

HTTP | FTP | SMTP | ... Camada de Aplicação
TCP/UDP Camada de Transporte
IP Camada de Rede
Vários Host para Camada de Rede

Figura 17.2 A pilha de protocolos utilizada por um protocolo da camada de aplicação como Hypertext Transfer Protocol.

Quando você utiliza o HTTP para transferir informações, o protocolo de HTTP chama o *Transmission Control Protocol (TCP),* que, por sua vez, conta com o *Internet Protocol (IP).* Esse protocolo, por sua vez, precisa de um protocolo apropriado para o hardware de rede sendo utilizado para pegar pacotes de dados e enviá-los como um sinal elétrico para nosso destino.

O HTTP é chamado protocolo de camada de aplicação. Há muitos outros protocolos da camada de aplicação como FTP, SMTP e telnet (como mostrado na figura) e outros como POP e IMAP. O TCP é um dos dois protocolos de camada de transporte utilizado em redes de TCP/IP. O IP é o protocolo na camada de rede. O host para camada de rede é responsável por conectar o host (computador) a uma rede. A pilha de protocolos de TCP/IP não especifica os protocolos utilizados para essa camada, uma vez que precisamos de protocolos diferentes para diferentes tipos de redes.

Ao enviar dados, estes são enviados para baixo pela pilha de uma aplicação até a mídia física da rede. Ao receber os dados, eles viajam da rede física, passando pela pilha, até a aplicação.

Utilizar o SSL adiciona uma camada transparente adicional a esse modelo. A camada de SSL fica entre a camada de transporte e a de aplicação. Isso é mostrado na Figura 17.3. A camada de SSL modifica os dados da aplicação de HTTP antes de fornecê-los à camada de transporte para enviá-los ao seu destino.

Figura 17.3 O SSL adiciona uma camada extra à pilha de protocolos bem como protocolos da camada de aplicação para controlar sua própria operação.

O SSL é teoricamente capaz de fornecer um ambiente de transmissão seguro para outros protocolos diferentes do HTTP, mas normalmente ele só é utilizado para HTTP. Outros protocolos podem ser utilizados porque a camada de SSL é essencialmente transparente. A camada de SSL fornece a mesma interface para protocolos acima dela que a camada de transporte subjacente. Então, essa camada lida de forma transparente com handshake, criptografia e decriptografia.

Quando um navegador Web conectar-se a um servidor seguro Web via HTTP, os dois precisam seguir um protocolo de handshake para concordar em itens como autenticação e criptografia.

A seqüência de handshake envolve os seguintes passos:

1. O navegador conecta-se a um servidor compatível com SSL e pede ao servidor para se autenticar.
2. O servidor envia seu certificado digital.
3. O servidor deve opcionalmente (e raramente) solicitar que o navegador se autentique.
4. O navegador apresenta uma lista dos algoritmos de criptografia e função de hash que ele suporta. O servidor seleciona a criptografia forte que ele também suporta.

5. O navegador e o servidor geram chaves de sessão:

   a. O navegador obtém a chave pública do servidor a partir do seu certificado digital e a utiliza para encriptar um número aleatoriamente gerado.

   b. O servidor responde enviando mais dados aleatórios em texto simples (a menos que o navegador tenha fornecido um certificado digital logo ao fazer a solicitação ao servidor, caso em que o servidor utilizará a chave pública do navegador).

   c. As chaves de criptografia para a sessão são geradas a partir desses dados aleatórios utilizando funções de hash.

Gerar dados aleatórios de boa qualidade, decriptar certificados digitais, gerar chaves e utilizar criptografia de chave pública leva tempo, então esse procedimento de handshake demora um pouco. Felizmente, os resultados são armazenados em cache; então, se o mesmo navegador e o mesmo servidor quiserem trocar diversas mensagens seguras, o processo de handshake e o tempo necessário de processamento só ocorrem uma vez.

Quando os dados são enviados por uma conexão de SSL, os seguintes passos ocorrem:

1. Eles são divididos em pacotes gerenciáveis.
2. Cada pacote é (opcionalmente) compactado.
3. Cada pacote tem um código de autenticação de mensagem (*message authentication code* – MAC) calculado utilizando um algoritmo de hashing.
4. O MAC e os dados compactados são combinados e encriptados.
5. Os pacotes encriptados são combinados com informações de cabeçalho e enviados à rede.

O processo inteiro é mostrado na Figura 17.4.

Figura 17.4 O SSL separa, compacta, faz o hashing e criptografa os dados antes de enviá-los.

Algo que você poderia notar a partir do diagrama é que o cabeçalho de TCP é adicionado depois que os dados são encriptados. Isso significa que informações de roteamento ainda poderiam ser potencialmente adulteradas e embora os *snoopers* não possam dizer quais informações estamos trocando, eles podem ver quem as está trocando.

A razão de o SSL incluir compactação antes da criptografia é que embora a maior parte de tráfego de rede possa ser (e freqüentemente é) compactada antes de ser transmitida através de uma rede,

os dados criptografados não compactam bem. Os esquemas de compactação contam com a identificação de repetição ou padrões nos dados. Tentar aplicar um algoritmo de compactação depois que os dados transformaram-se em um arranjo de bits efetivamente aleatório via criptografia normalmente não faz sentido. Seria desagradável se o SSL, que foi projetado para aumentar segurança de rede, tivesse o efeito colateral de aumentar drasticamente o tráfego de rede.

Embora o SSL seja relativamente complexo, os usuários e desenvolvedores ficam isolados da maior parte do que ocorre, uma vez que suas interfaces externas simulam protocolos existentes.

No futuro, o SSL 3.0 talvez seja substituído pelo TLS 1.0 (Transport Layer Security), mas na época em que este livro era escrito, o TLS era um padrão ainda experimental e não era suportado por nenhum servidor ou navegador. O TLS é projetado para ser um padrão verdadeiramente aberto, em vez de um padrão definido por uma organização e disponibilizado para outras. Ele é baseado diretamente no SSL 3.0, mas contém aprimoramentos projetados para superar as fraquezas do SSL.

Verificando a entrada do usuário

Um dos princípios de construção de uma aplicação Web segura é que você nunca deve confiar na entrada do usuário. Sempre verifique os dados do usuário antes de colocá-los em um arquivo ou banco de dados ou enviá-los através de um comando executável do sistema.

Em várias passagens deste livro, discutimos técnicas que você pode utilizar para verificar a entrada do usuário. Listaremos brevemente algumas delas aqui como uma referência.

- A função `addslashes( )` deve ser utilizada para filtrar dados do usuário antes de eles serem passados para um banco de dados. Essa função "escapará" caracteres que talvez sejam problemáticos para um banco de dados. Você pode utilizar a função `stripslashes( )` para retornar os dados à sua forma original.
- Você pode ativar ou desativar as diretivas `magic_quotes_gpc` e `magic_quotes_runtime` no arquivo `php.ini`. Essas diretivas irão adicionar ou remover automaticamente as barras para você. A diretiva `magic_quotes_gpc` aplicará essa formatação a variáveis entrantes de `GET`, `POST` e cookie, e a diretiva `magic_quote_runtime` aplicará essa formatação a dados que saem e entram no banco de dados.
- A função `escapeshellcmd( )` deve ser utilizada ao passar dados de usuário para uma chamada a `system( )` ou `exec( )` ou para sinais de crase. Isso eliminará qualquer metacaractere que possa ser utilizado para forçar seu sistema a executar comandos arbitrários inseridos por um usuário malicioso.
- Você pode utilizar a função `strip_tags( )` para eliminar tags de HTML e tags de PHP de uma string. Isso evitará que usuários plantem scripts maliciosos nos dados do usuário que você poderia ecoar de volta para o navegador.
- Você pode utilizar a função `htmlspecialchars( )`, que converterá caracteres em suas entidades de HTML equivalentes. Por exemplo, < será convertido em `<`. Isso converterá qualquer tag de script em caracteres inofensivos.

Fornecendo armazenamento seguro

Os três diferentes tipos de dados armazenados (arquivos de HTML ou de PHP, dados relacionados a script e dados do MySQL) serão geralmente armazenados em áreas diferentes do mesmo disco, mas são mostrados separadamente na Figura 17.1. Cada tipo de armazenamento exige precauções diferentes e será examinado separadamente.

O tipo de dados mais perigoso que armazenamos é conteúdo executável. Em um Web site, isso normalmente significa scripts. Você precisa ser muito cuidadoso para que suas permissões de arquivo sejam configuradas corretamente dentro da hierarquia Web. Com isso queremos dizer a árvore

de diretórios iniciando a partir de htdocs em um servidor Apache ou iniciando a partir de inetpub em um servidor IIS. Outros precisam ter permissão para ler os scripts a fim de verem a saída, mas não devem ser capazes de gravar ou editar os scripts.

A mesma condição se aplica a diretórios dentro da hierarquia Web. Só devemos ser capazes de gravar nesses diretórios. Outros usuários, incluindo o usuário sob o qual o servidor Web executa, não devem ter permissão de gravar nem de criar novos arquivos em diretórios que possam ser carregados a partir do servidor Web. Se você permitir que outros gravem arquivos aqui, eles poderiam escrever um script malicioso e executá-lo carregando-o pelo servidor Web.

Se seus scripts precisam de permissão para gravar em arquivos, crie um diretório fora da árvore Web para esse propósito. Isso se aplica particularmente a scripts que carregam arquivos. Os scripts e os dados que eles gravam não devem se misturar.

Ao gravar informações sigilosas, você talvez seja tentado a encriptá-las primeiro. Mas, em geral, essa abordagem é de pouco valor. Vamos colocar isso desta maneira: se você tiver um arquivo chamado creditcardnumbers.txt no seu servidor Web e um cracker tiver acesso ao servidor e ler esse arquivo, o que mais ele pode ler? A fim de encriptar e decriptar os dados, você precisará de um programa para encriptar dados, um programa para decriptar dados e um ou mais arquivos-chave. Se o cracker puder ler os dados, provavelmente nada o impedirá de ler sua chave e outros arquivos.

Encriptar os dados poderia ser valioso para um servidor Web, mas somente se o software e a chave para decriptar os dados não forem armazenados no servidor Web, mas somente existirem em outra máquina. Uma maneira de lidar seguramente com informações sigilosas seria encriptá-los no servidor e então transmiti-los para outra máquina, talvez via e-mail.

Os dados de banco de dados são semelhantes a arquivos de dados. Se você configurar o MySQL corretamente, somente ele pode gravar nos arquivos de dados. Isso significa que só precisamos nos preocupar com acessos de usuários a partir de dentro do MySQL. Já discutimos o próprio sistema de permissões do MySQL, que atribui direitos particulares a nomes de usuário particulares em hosts particulares.

Uma coisa que merece atenção especial é que você freqüentemente precisará escrever uma senha do MySQL em um script de PHP. Seus scripts de PHP são em geral publicamente carregáveis. Isso não é um desastre como talvez pareça a princípio. A menos que a configuração de servidor Web seja quebrada, seu código-fonte em PHP não será visível do lado de fora.

Se o servidor Web estiver configurado para analisar sintaticamente arquivos com a extensão .php utilizando o interpretador de PHP, os forasteiros não serão capazes de visualizar os códigos-fonte não-interpretados. Entretanto, você deve ter cuidado ao utilizar outras extensões. Se você colocar arquivos .inc em seus diretórios Web, qualquer pessoa que os solicitar receberá o código-fonte não analisado sintaticamente. Você precisa colocar arquivos include fora da árvore Web, configure seu servidor para não entregar arquivos com essa extensão ou utilizar também .php como a extensão nesses arquivos.

Se você estiver compartilhando um servidor Web com outros, sua senha do MySQL talvez seja visível a outros usuários na mesma máquina que também possam executar scripts via mesmo servidor Web. Dependendo de como seu sistema estiver configurado, isso talvez seja inevitável. Isso pode ser evitado configurando um servidor Web para executar scripts como usuários individuais ou fazendo com que cada usuário execute sua própria instância do servidor Web. Se você não for o administrador do seu servidor Web (como é provável se você estiver compartilhando um servidor), talvez seja válido discutir isso com seu administrador e explorar as opções de segurança.

Determinando por que você está armazenando números de cartão de crédito

Ao discutir armazenamento seguro para dados sigilosos, um tipo de dado sigiloso merece menção especial. Os usuários da Internet são paranóicos sobre seus números de cartão de crédito. Se você for armazená-los, precisará ter muito cuidado. Mas antes você precisa se perguntar por que vai fazer isso e se é realmente necessário.

O que você vai fazer com um número de cartão? Se tiver de processar uma transação que não se repetirá e tiver processamento de cartão em tempo real, seria melhor aceitar o número de cartão de seu cliente e enviá-lo diretamente para seu gateway de processamento de transação sem armazená-lo.

Se você tiver de fazer cobranças periódicas, como cobrar de alguém uma taxa mensal no mesmo cartão para uma assinatura periódica, isso talvez não seja uma opção. Nesse caso, você deve pensar em armazenar os números em algum outro lugar diferente do servidor Web.

Se você vai armazenar enormes variedades de detalhes do cartão dos seus clientes, certifique-se de que tem um administrador de sistema qualificado e um pouco paranóico que tenha tempo suficiente de verificar fontes atualizadas de informações de segurança para o sistema operacional e outros produtos que você utiliza.

Utilizando criptografia em PHP

Uma tarefa simples, mas útil, que podemos utilizar para demonstrar a criptografia é enviar e-mail encriptado. O padrão *de fato* para e-mail encriptado foi o PGP durante muitos anos, que significa Pretty Good Privacy. Philip R. Zimmermann escreveu o PGP especificamente para agregar privacidade ao e-mail.

Versões freeware do PGP estão disponíveis, mas você deve notar que isso não é software gratuito (Free Software). A versão freeware só pode ser legalmente utilizada para uso não-comercial.

Se você for um cidadão norte-americano nos Estados Unidos ou um cidadão canadense no Canadá, pode obter a versão freeware em http://web.mit.edu/network/pgp.html.

Se quiser utilizar o PGP para uso comercial e estiver nos Estados Unidos ou Canadá, você pode obter uma licença comercial da Network Associates. Consulte http://www.pgp.com para obter detalhes.

Para obter o PGP para utilização fora dos Estados Unidos e do Canadá, veja a lista de sites de download internacional na página internacional do PGP: http://www.pgpi.org.

Uma alternativa da Open Source ao PGP tornou-se recentemente disponível. O GPG – Gnu Privacy Guard – é um substituto gratuito e livre para o PGP. Ele contém algoritmos não-patenteados e pode ser utilizado comercialmente sem restrição.

Os dois produtos realizam a mesma tarefa de maneiras relativamente semelhantes. Talvez não importe se você pretende utilizar as ferramentas de linha de comando, mas cada um tem outras interfaces úteis como plug-ins para programas de e-mail que automaticamente decriptarão o e-mail quando ele for recebido.

O GPG está disponível em http://www.gnupg.org.

Você pode utilizar os dois produtos juntos, criando uma mensagem encriptada utilizando o GPG para alguém que utiliza o PGP (contanto que seja uma versão recente) para decriptar. Como estamos interessados na criação de mensagens no servidor Web, forneceremos um exemplo aqui utilizando o GPG. Utilizar o PGP, por sua vez, não exigirá muitas alterações.

Assim como os requisitos normais para os exemplos neste livro, você precisará ter o GPG disponível para esse código funcionar. O GPG talvez já esteja instalado no sistema. Se não estiver, não tem importância: o procedimento de instalação é muito simples e direto, mas a configuração pode ser um pouco difícil.

Instalando o GPG

Para adicionar GPG à nossa máquina Linux, fizemos o download do repositório de arquivos apropriado em www.gnupg.org. Dependendo de você escolher o repositório de arquivos `.tar.gz` ou `.tar.bz2`, precisará utilizar `gunzip` e `tar` ou extrair os arquivos do repositório de arquivos.

Para compilar e instalar o programa, utilize os mesmos comandos utilizados para a maioria dos programas Linux:

`configure` (ou `./configure` dependendo de seu sistema)

```
make
make install
```

Se não for o usuário root, você precisará executar o script configure com a opção --prefix da seguinte maneira:

```
./configure --prefix=/path/to/your/directory
```

Isso porque um usuário não-root não terá acesso ao diretório padrão do GPG.

Se tudo correr bem, o GPG será compilado e o executável copiado para /usr/local/bin/gpg ou o diretório que você especificou. Você pode alterar muitas opções. Veja a documentação do GPG para detalhes.

Para um servidor Windows, o processo é até mais fácil. Faça download do arquivo zip, descompacte-o e coloque o gpg.exe em algum lugar no PATH. (C:\Windows\ ou semelhante). Crie um diretório em C:\gnupg. Abra um prompt de comando e digite **gpg**.

Você também precisa instalar o GPG ou o PGP e gerar um par de chaves no sistema a partir do qual você planeja verificar correio.

No servidor Web, há bem poucas diferenças entre as versões de linha de comando do GPG e o PGP, então também poderíamos utilizar o GPG uma vez que ele é gratuito. Na máquina a partir da qual você lê o correio, talvez prefira comprar uma versão comercial do PGP para ter um plug-in de interface gráfica com o usuário mais interessante para o leitor de correio.

Se não tiver um ainda, gere um par de chaves na sua máquina de leitura de correio. Lembre-se de que um par de chaves consiste em uma chave pública que outras pessoas (e seu script de PHP) utilizam para encriptar correio antes de enviá-lo para você e uma chave privada, que você utiliza para decriptar mensagens recebidas ou assinar correio enviado. É importante que a geração de chave seja feita na máquina de leitura de correio, em vez de no servidor Web, já que a chave privada não deve ser armazenada no servidor Web.

Se você estiver utilizando a versão de linha de comando do GPG para gerar suas chaves, insira o seguinte comando:

```
gpg --gen-key
```

Serão feitas várias perguntas a você. A maioria delas tem uma resposta padrão que pode ser aceita. Em linhas separadas, será solicitado a você seu nome real, seu endereço de e-mail e um comentário, que será utilizado para nomear a chave. (Minha chave tem o nome 'Luke Welling <luke@tangledweb.com.au>'. Certamente você pode distinguir o padrão. Se eu também tivesse fornecido um comentário, ele estaria entre o nome e o endereço.)

Para exportar a chave pública a partir do novo par de chaves, você pode utilizar o comando:

```
gpg --export > nomedoarquivo
```

Isso fornecerá um arquivo binário adequado para importar para dentro do chaveiro do GPG ou do PGP em outra máquina. Se você quiser enviar essa chave por e-mail às pessoas, para que elas possam importar para seus chaveiros, você pode criar uma versão de ASCII assim:

```
gpg --export --a > nomedoarquivo
```

Tendo extraído a chave pública, você pode carregar o arquivo para sua conta no servidor Web. Você pode fazer isso com FTP.

Os seguintes comandos assumem que você está utilizando o UNIX. Os passos são os mesmos utilizados para o Windows, mas nomes de diretório e comandos de sistema serão diferentes. Efetue logon na sua conta no servidor Web e altere as permissões no arquivo para que os outros usuários sejam capazes de ler. Digite:

```
chmod 644 nomedoarquivo
```

Você precisará criar um chaveiro de modo que o usuário com que seus scripts de PHP são executados possa utilizar o GPG. Que usuário é esse depende da maneira como o servidor está configurado. Freqüentemente é o usuário 'nobody', mas poderia ser qualquer outro.

Mude para ser o usuário do servidor Web. Você precisará ter acesso root para o servidor fazer isso. Em muitos sistemas, o servidor Web executa como nobody. Os exemplos a seguir assumem isso. (Você pode alterá-lo para o usuário apropriado no seu sistema.) Se esse for o caso no seu sistema, digite:

```
su root
su nobody
```

Crie um diretório para nobody para armazenar seu chaveiro e outras informações de configuração do GPG. Isso precisará estar no diretório inicial (ou *home directory*) de nobody.

O diretório inicial de cada usuário é especificado em /etc/passwd. Em muitos sistemas Linux, o diretório inicial de nobody é por padrão /, no qual nobody não terá permissão para gravar. Em muitos sistemas BSD, o diretório inicial de nobody é por padrão /nonexistent, o qual, por não existir, não pode ser gravado. Em nosso sistema, nobody foi atribuído ao diretório inicial /tmp. Você precisará certificar-se de que os usuários do seu servidor Web têm um diretório inicial em que podem gravar.

Digite:

```
cd ~
mkdir .gnupg
```

O usuário nobody precisará de sua própria chave de assinatura. Para criar isso, execute este comando novamente:

```
gpg --gen-key
```

Uma vez que seu usuário nobody provavelmente recebe muito pouco e-mail pessoal, você pode criar uma chave de sign-in somente (*signing only key*) para ele. O único propósito dessa chave é permitir confiar na chave pública que extraímos anteriormente.

Para importar a chave pública que exportamos anteriormente, utilize o seguinte:

```
gpg --import nomedoarquivo
```

Para informar o GPG que queremos confiar nessa chave, precisamos editar as propriedades da chave utilizando:

```
gpg --edit-key 'Luke Welling <luke@tangledweb.com.au>'
```

Nessa linha, o texto entre aspas é o nome da chave. Obviamente, o nome de sua chave não será 'Luke Welling <luke@tangledweb.com.au>', mas uma combinação do nome, comentário e endereço de e-mail que você forneceu ao gerá-la.

As opções dentro desse programa incluem help, que descreverá os comandos disponíveis – trust, sign, e save.

Digite **trust** e informe o GPG que você confia completamente na sua chave. Digite **sign** para assinar essa chave pública utilizando a chave privada nobody. Por fim, digite **save** para sair desse programa, mantendo suas alterações.

Testando GPG

O GPG deve estar agora configurado e pronto para utilização.

Criar um arquivo contendo algum texto e salvá-lo como test.txt permitirá testá-lo.

Digitar o comando a seguir (modificado para usar o nome de sua chave)

```
gpg -a --recipient 'Luke Welling <luke@tangledweb.com.au>' --encrypt test.txt
```

deve fornecer-lhe o aviso

```
gpg: Warning: using insecure memory!
```

e criar um nome de arquivo `test.txt.asc`. Se abrir `test.txt.asc` você deve ver uma mensagem encriptada assim:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org

hQEOAODU7hVGgdtnEAQAhr4HgR7xpIBsK9CiELQw85+k1QdQ+p/FzqL8tICrQ+B3
OGJTEehPUDErwqUw/uQLTdsOrloPSrIAZ7c6GVkhOYEVBj2MskT81IIBvdo95OyH
K9PUCvg/rLxJ1kxe4Vp8QFET5E3FdII/ly8VP5gSTE7gAgmOSbFf3S91PqwMyTkD
/2oJEvL6e3cP384sOi8lrBbDbOUAAhCjjXt2DX/uX9q6P18QW56UICUOn4DPaW1G
/gnNZCkcVDgLcKfBjbkB/TCWWhpA7o7kX4CIcIh7KlIMHY4RKdnCWQf271oE+8i9
cJRSCMsFIoI6MMNRCQHY6p9bfxL2uE39IRJrQbe6xoEeOnkBOuTYxiLOTG+FrNrE
tvBVMSOnsHu7HJey+oY4Z833pk5+MeVwYumJwlvHjdZxZmV6wz46GO2XGT17b28V
wSBnWOoBHSZsPvkQXHTOq65EixP8y+YJvBN3z4pzdHOXa+NpqbH7q3+xXmd3OhDR
+u7t6MxTLDbgC+NR
=gfQu
-----END PGP MESSAGE-----
```

Você deve ser capaz de transferir esse arquivo para o sistema em que inicialmente gerou a chave e executou

```
gpg test.txt.asc
```

para recuperar o texto original. O texto será gravado em um arquivo com o mesmo nome que tinha antes – nesse caso, `test.txt`.

Para que o texto seja ecoado para a tela, use o flag `-d`:

```
gpg -d test.txt.asc
```

Para colocar o texto em um arquivo de sua escolha em vez do nome default, você pode usar o flag `-o` assim como especificar um arquivo de saída como este:

```
gpg --do test.out test.txt.asc
```

Observe que o arquivo de saída é nomeado primeiro.

Se tiver o GPG configurado de modo que o usuário em que foram executados os scripts PHP possa utilizá-lo a partir da linha de comando, você está quase lá. Se isso não funcionar, consulte o administrador de seu sistema ou a documentação GPG.

As Listagens 17.1 e 17.2 permitem que as pessoas enviem e-mail criptografado utilizando o PHP para chamar o GPG.

Listagem 17.1 **`private_mail.php`** **– Nosso formulário HTML para enviar e-mail encriptado**

```
<html>
<body>
<h1>Send Me Private Mail</h1>

<?php
 // Talvez você precise alterar essa linha, se não utilizar
 // as portas padrão, 80 para tráfego normal e 443 para SSL
 if($HTTP_SERVER_VARS['SERVER_PORT']!=443)
   echo '<p><font color="red">
          WARNING: you have not connected to this page using SSL.
          Your message could be read by others.</font></p>';
?>

<form method="post" action="send_private_mail.php"><br />
Your email address:<br />
```

Listagem 17.1 Continuação

```
<input type="text" name="from" size"="38"><br />
Subject:<br />
<input type="text" name="title" size="38"><br />
Your message:<<br />
<textarea name="body" cols="30" rows="10">
</textarea><br />
<input type="submit" value="Send!">
</form>
</body>
</html>
```

Listagem 17.2 **send_private_mail.php** – Nosso script de PHP para chamar o GPG e enviar e-mail encriptado

```
<?php
  //cria nome de variável abreviado  $from = $HTTP_POST_VARS['from'];
  $title = $HTTP_POST_VARS['title'];
  $body = $HTTP_POST_VARS['body'];

  $to_email = 'luke@localhost';

  // Diz ao gpg onde encontrar o chaveiro
  // Nesse sistema, o diretório inicial do usuário nobody é /tmp/
  putenv('GNUPGHOME=/tmp/.gnupg');

  //cria um nome de arquivo único
  $infile = tempnam('', 'pgp');
  $outfile = $infile.'.asc';

  //grava o texto do usuário em um arquivo
  $fp = fopen($infile, 'w');
  fwrite($fp, $body);
  fclose($fp);

  //configura nosso comando
  $command =  "/usr/local/bin/gpg -a \\
              --recipient 'Luke Welling <luke@tangledweb.com.au>' \\
              --encrypt -o $outfile $infile";

  // executa nosso comando gpg
  system($command, $result);

  //exclui o arquivo temporário não-encriptado
  unlink($infile);

  if($result==0)
  {
    $fp = fopen($outfile, 'r');
    if(!$fp||filesize ($outfile)==0)
    {
      $result = -1;
    }
    else
    {
      //lê o arquivo encriptado
      $contents = fread ($fp, filesize ($outfile));
      //exclui o arquivo temporário encriptado
      unlink($outfile);
```

Listagem 17.2 Continuação

```
      mail($to_email, $title, $contents, "From: $from\n");
      echo '<h1>Message Sent</h1>
            <p>Your message was encrypted and sent.</p>
            <p>Thank you.'</p>';
    }
  }

  if($result!=0)
  {
    echo '<h1>Error:</h1>
          <p>Your message could not be encrypted, so has not been sent.</p>
          <p>Sorry.</p>';
  }
?>
```

Para fazer esse código funcionar, você precisará alterá-lo um pouco. O e-mail será enviado para o endereço em `$to_email`.

A linha

```
putenv('GNUPGHOME=/tmp/.gnupg');
```

precisará ser alterada para refletir a localização de seu chaveiro de GPG. Em nosso sistema, o servidor Web executa como o usuário `nobody` e tem o diretório inicial `/tmp/`.

Estamos utilizando a função tempnam( ) para criar um nome de arquivo temporário único. Você pode especificar tanto o diretório como um prefixo de nome de arquivo. Vamos criar e excluir esses arquivos quase imediatamente, então não é muito importante o nome que lhe atribuímos. Estamos especificando um prefixo `'pgp'`, mas permitindo que o PHP utilize o diretório temporário do sistema.

A instrução

```
$command =  '/usr/local/bin/gpg -a '.
            '--recipient 'Luke Welling <luke@tangledweb.com.au>' '.
            '--encrypt -o $outfile $infile';
```

configura o comando e os parâmetros que serão utilizados para chamar o GPG. Ela precisará ser modificada para atender às suas necessidades. Como na ocasião em que o utilizamos na linha de comando, você precisa dizer ao GPG qual chave utilizar para encriptar a mensagem.

A instrução

```
system($command, $result);
```

executa as instruções armazenadas em `$command` e armazena o valor de retorno em `$result`. Poderíamos ignorar o valor de retorno, mas ele permite ter uma instrução `if` e diz ao usuário que algo saiu errado.

Quando concluirmos os arquivos temporários que utilizamos, excluímos esses arquivos utilizando a função `unlink( )`. Isso significa que o e-mail não-encriptado do nosso usuário está sendo armazenado no servidor por um breve período. Se o servidor falhar durante a execução, ainda é possível que o arquivo permaneça no servidor.

Enquanto estamos pensando na segurança de nosso script, é importante considerar todos os fluxos de informações dentro de nosso sistema. O GPG encriptará nosso e-mail e permitirá ao nosso destinatário decriptá-lo, mas como as informações chegam originalmente do remetente? Se estivermos fornecendo uma interface Web para enviar correio de GPG encriptado, o fluxo das informações será parecido com a Figura 17.5.

Nessa figura, cada seta representa nossa mensagem sendo enviada de uma máquina para outra. Toda vez que a mensagem é enviada, ela viaja pela Internet e talvez passe por várias redes e máquinas intermediárias.

Navegador de quem envia → 1 → Servidor Web → 2 → Servidor de correio do destinatário → 3 → Cliente do correio do destinatário

Figura 17.5 Em nossa aplicação de e-mail encriptado, a mensagem é enviada via Internet três vezes.

O script que estamos examinando aqui existe na máquina rotulada como Servidor Web no diagrama. No servidor Web, a mensagem será encriptada utilizando a chave pública do destinatário. Então, ela será enviada via SMTP para o servidor de correio do destinatário. O destinatário conecta-se ao seu servidor de correio, utilizando provavelmente POP ou IMAP e faz download da mensagem utilizando um leitor de correio. Aqui ele decriptará a mensagem utilizando sua chave privada.

As transferências de dados na Figura 17.5 são rotuladas 1, 2 e 3. Para as etapas 2 e 3, as informações sendo transmitidas são uma mensagem GPG encriptada e são de pouco valor a qualquer pessoa que não tenha a chave privada. Para a transferência 1, a mensagem sendo transmitida é o texto que o remetente inseriu no formulário.

Se nossas informações são suficientemente importantes que precisamos encriptá-las durante a segunda e terceira etapas de sua viagem, é um pouco ridículo enviá-las decriptadas para a primeira etapa. Portanto, esse script pertence a um servidor que utiliza SSL.

Se conectarmos nosso script utilizando uma porta diferente da 443, ele fornecerá um aviso. Essa é a porta padrão para SSL. Verificamos isso verificando o valor de `$_SERVER['SERVER_PORT']`. As conexões SSl entram na porta 443. Qualquer outra conexão vai causar erro.

Em vez de fornecer uma mensagem de erro, poderíamos lidar com essa situação de outras maneiras. Poderíamos redirecionar o usuário para o mesmo URL via uma conexão de SSL. Também poderíamos escolher ignorar esse erro porque não é normalmente importante se o formulário foi entregue utilizando uma conexão segura. O que normalmente é importante é que os detalhes que o usuário digitou no formulário sejam enviados para nós seguramente. Simplesmente poderíamos ter fornecido um URL completo como a ação de nosso formulário.

Atualmente, a forma aberta de nossa tag é semelhante a:

```
<form method="post" action="send_private_mail.php">
```

Poderíamos alterá-la para enviar dados via SSL mesmo que o usuário se conectasse sem SSL, como a seguir:

```
<form method="post" action="https://webserver/send_private_mail.php">
```

Se codificarmos o URL completo dessa maneira, podemos ter garantia de que os dados dos visitantes serão enviados utilizando SSL, mas precisaremos modificar o código toda vez que o utilizarmos em outro servidor ou mesmo em outro diretório.

Embora nesse caso e em muitos outros, não seja importante que o formulário vazio seja enviado para o usuário via SSL, normalmente é uma boa idéia fazer isso. Ver o símbolo do pequeno cadeado na barra de status de seus navegadores tranqüiliza as pessoas de que as suas informações serão enviadas seguramente. Elas não devem precisar ver sua origem HTML para descobrir qual é o atributo do comando action do formulário.

Leitura adicional

A especificação para versões 3.0 do SSL está disponível no Netscape:

http://home.netscape.com/eng/ssl3/

Se quiser conhecer mais sobre como funcionam as redes e os protocolos de rede, um texto introdutório clássico é *Redes de Computadores* de Andrew S. Tannenbaum, publicado pela Campus.

A seguir

Isso encerra nossa discussão sobre comércio eletrônico e questões de segurança. Na próxima seção, veremos algumas técnicas mais avançadas de PHP incluindo como interagir com outras máquinas na Internet, gerar imagens instantaneamente e utilizar controle de sessão.

IV

Técnicas avançadas de PHP

18

Interagindo com o sistema de arquivos e o servidor

NO CAPÍTULO 2, VIMOS COMO LER DADOS DE E GRAVAR DADOS em arquivos no servidor Web. Neste capítulo, abordaremos outras funções do PHP que permitem interagir com o sistema de arquivos no servidor Web.

Os tópicos-chave abordados neste capítulo incluem:

- Carregando arquivos com o PHP;
- Utilizando as funções de diretório;
- Interagindo com arquivos no servidor;
- Executando programas no servidor;
- Utilizando variáveis de ambiente do servidor.

Para discutir o uso dessas funções, veremos um exemplo.

Considere uma situação em que você gostaria que seu cliente fosse capaz de atualizar algum conteúdo do Web site – por exemplo, as notícias atuais sobre sua empresa. (Ou talvez você queira uma interface mais amigável que FTP para você mesmo.) Uma abordagem para isso é deixar o cliente carregar os arquivos de conteúdo como texto simples. Esses arquivos então estarão disponíveis no site, por um modelo que você projetou com o PHP, como fizemos no Capítulo 6.

Antes de mergulharmos nas funções de sistema de arquivos, vamos examinar brevemente como funciona o upload de arquivo.

Introdução ao upload de arquivo

Uma parte muito útil da funcionalidade do PHP é suporte para upload de HTTP. Em vez de arquivos virem do servidor para o navegador utilizar o HTTP, eles vêm na direção oposta, isto é, do navegador para o servidor. Normalmente, isso é implementado com uma interface de formulário HTML. O formulário que utilizaremos em nosso exemplo é mostrado na Figura 18.1.

Como você pode ver, o formulário tem uma caixa em que o usuário pode inserir um nome de arquivo ou clicar no botão Browse para navegar arquivos disponíveis localmente. Talvez você não tenha visto um formulário de upload de arquivo antes. Veremos como implementar isso em um instante.

Depois que um nome de arquivo foi inserido, o usuário pode clicar em Send File e o arquivo será carregado para o servidor, onde há um script de PHP esperando por ele.

Figura 18.1 O formulário de HTML que utilizamos para upload de arquivo tem campos e tipos de campos diferentes dos tipos de um formulário normal de HTML.

HTML para upload de arquivo

A fim de implementar o upload de arquivo, precisamos utilizar alguma sintaxe de HTML que existe especialmente para esse propósito. A HTML para esse formulário é mostrada na Listagem 18.1.

Listagem 18.1 **upload.html – Formulário HTML para upload de arquivo**

```
<html>
<head>
  <title>Administration - upload new files</title>
</head>
<body>
<h1>Upload new news files</h1>
<form enctype="multipart/form-data" action="upload.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
  Upload this file: <input name="userfile" type="file">
  <input type="submit" value="Send File">
</form>
</body>
</html>
```

Observe que esse formulário utiliza POST. O upload de arquivo também trabalhará com o método PUT suportado pelo Netscape Composer e Amaya. Eles não funcionarão com o GET.

Os recursos extras nesse formulário são:

- Na tag <form>, você deve configurar o atributo enctype="multipart/form-data" para permitir que o servidor saiba que um arquivo está vindo junto com as informações regulares de formulário.

- Você deve ter um campo de formulário que configure o tamanho máximo do arquivo que pode ser carregado. Esse é um campo oculto e é mostrado aqui como:

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
```

  O nome desse campo de formulário deve ser MAX_FILE_SIZE. O valor é o tamanho máximo (em bytes) de arquivos que você permitirá que as pessoas carreguem. No momento, configuramos isso como 1.000.000 bytes (aproximadamente um megabyte). Você talvez deseje torná-lo maior ou menor para sua aplicação.

- Você precisa de uma entrada de type file, mostrada aqui como:

```
<input name="userfile" type="file">
```

Você pode escolher o nome que quiser para o arquivo, mas não o esqueça já que utilizará esse nome para acessar seu arquivo a partir do script de PHP receptor.

Uma nota sobre segurança

Antes de prosseguirmos, vale observar que algumas versões do PHP apresentaram vulnerabilidades de segurança no código de upload de arquivo. Se decidir utilizar o upload de arquivo no servidor de produção, você deverá certificar-se de que está utilizando a versão mais atual do PHP e prestar atenção aos patches.

Isso não deve impedir que você utilize essa tecnologia tão útil, mas você deve ter cuidado com a maneira como escreve seu código e pensar em restringir o acesso aos uploads de arquivos a, por exemplo, administradores de site e gerenciadores de conteúdo.

Escrevendo o PHP para lidar com o arquivo

Escrever o PHP para capturar o arquivo é bastante simples e direto, mas depende da versão do PHP e das definições de configuração. Os nomes de função e variáveis mudaram ao longo das diferentes versões e dependem se você tem register_globals ativado. O código aqui não requer register_globals, mas requer pelo menos a versão 4.1 do PHP.

Quando o arquivo é carregado, ele vai rapidamente para um local temporário no servidor Web. Por default, esse é o diretório temporário principal do servidor Web. Se você não mover, copiar ou renomear o arquivo antes de o script terminar a execução, ele será excluído quando o script terminar.

Os dados que você precisa manipular no script PHP são armazenados no array superglobal $_FILES. Se register_globals estiver ativado, você também pode acessar as informações diretamente pelos nomes das variáveis. Entretanto, provavelmente essa é uma área em que é mais importante ter register_globals desativado, ou, pelo menos, agir como se estivesse, usar o array superglobal e ignorar os globais.

As entradas em $_FILES serão armazenadas com o nome da tag <file> do formulário HTML. O elemento do formulário é chamado userfile, então o array terá os seguintes conteúdos:

- O valor armazenado em $_FILE['userfile']['tmp_name'] ou $HTTP_POST_FILES['userfile']['tmp_name'] é onde o arquivo foi temporariamente armazenado no servidor Web.
- O valor armazenado em $_FILE['userfile']['name'] ou $HTTP_POST_FILES['userfile']['name'] é o nome de arquivo no sistema do usuário.
- O valor armazenado em $_FILE['userfile']['size'] ou $HTTP_POST_FILES['userfile']['size'] é o tamanho do arquivo em bytes.
- O valor armazenado em $_FILE['userfile']['type'] ou $HTTP_POST_FILES['userfile']['type'] é o tipo MIME do arquivo, por exemplo, text/plain ou image/gif.
- O valor armazenado em $_FILE['userfile']['error'] ou $HTTP_POST_FILES['userfile']['error'] lhe dará quaisquer códigos de erro associados ao upload do arquivo. Isso foi adicionado ao PHP 4.2.0.

Considerando que você sabe onde está o arquivo e qual é o nome dele, agora você pode copiá-lo em algum lugar útil. No final da execução do script, o arquivo temporário será excluído. Portanto, você deve mover ou renomear o arquivo se quiser mantê-lo.

Em nosso exemplo, utilizaremos os arquivos carregados como artigos de notícias recentes, então cortaremos qualquer tag em que eles possam estar e movê-las para um diretório mais útil. Um script que faz isso é mostrado na Listagem 18.2.

Listagem 18.2 **upload.php** – PHP para capturar os arquivos do formulário HTML

```
<html>
<head>
 <title>Uploading...</title>
</head>
<body>
<h1>Uploading file...</h1>
<?php

 if ($_FILES['userfile']['error'] > 0)
 {
   echo 'Problem: ';
   switch ($_FILES['userfile']['error'])
   {
     case 1:  echo 'File exceeded upload_max_filesize';  break;
     case 2:  echo 'File exceeded max_file_size';  break;
     case 3:  echo 'File only partially uploaded';  break;
     case 4:  echo 'No file uploaded';  break;
   }
   exit;
 }

 // O arquivo possui o tipo MIME correto?
 if ($_FILES['userfile']['type'] != 'text/plain')
 {
   echo 'Problem: file is not plain text';
   exit;
 }

 // insere arquivo onde gostaríamos
 $upfile = '/uploads/'.$_FILES['userfile']['name'];

 if (is_uploaded_file($_FILES['userfile']['tmp_name']))
 {
    if (!move_uploaded_file($_FILES['userfile']['tmp_name'], $upfile))
    {
       echo 'Problem: Could not move file to destination directory';
       exit;
    }
 }
 else
 {
   echo 'Problem: Possible file upload attack. Filename: ';
   echo $_FILES['userfile']['name'];
   exit;
 }

 echo 'File uploaded successfully<br><br>';

 // reformata o conteúdo do arquivo
 $fp = fopen($upfile, 'r');
 $contents = fread ($fp, filesize ($upfile));
 fclose ($fp);

 $contents = strip_tags($contents);
 $fp = fopen($upfile, 'w');
 fwrite($fp, $contents);
 fclose($fp);

 // mostra o que foi carregado
 echo 'Preview of uploaded file contents:<br><hr>';
```

Listagem 18.2 **Continuação**

```
echo $contents;
echo '<br><hr>';
?>
</body>
</html>
```

Curiosamente, a maior parte desse script é verificação de erros. O upload de arquivo envolve riscos potenciais de segurança e precisamos amenizar esses riscos onde for possível. Precisamos validar o arquivo carregado com o máximo cuidado possível para nos certificar de que é seguro ecoar para nossos visitantes.

Vamos passar pelas partes principais do script. Começamos a verificar o código de erro retornado por `$HTTP_POST_FILES['userfile']['error']`. Esse código de erro foi introduzido no PHP 4.2.0. A partir do PHP 4.3, também há uma constante associada a cada um dos códigos. Os possíveis valores e constantes são:

- `UPLOAD_ERROR_OK`, valor 0, significa que não ocorreu nenhum erro.
- `UPLOAD_ERR_INI_SIZE`, valor 1, significa que o tamanho do arquivo de upload excede o valor máximo especificado no arquivo `php.ini` com a diretiva `upload_max_ filesize`.
- `UPLOAD_ERR_FORM_SIZE`, valor 2, significa que o tamanho do arquivo de upload excede o valor máximo especificado no formulário HTML no elemento `MAX_FILE_SIZE`.
- `UPLOAD_ERR_PARTIAL`, valor 3, significa que o upload do arquivo foi apenas parcial.
- `UPLOAD_ERR_NO_FILE`, valor 4, significa que nenhum upload de arquivo foi feito.

Se você quiser utilizar uma versão mais antiga do PHP, pode realizar uma versão manual de algumas dessas verificações utilizando código de exemplo do manual do PHP ou das edições mais antigas deste livro.

Você também pode verificar o tipo MIME. Nesse caso, queremos que você carregue apenas os arquivos de texto, então teste o tipo MIME assegurando que `$_FILES['userfile']['type']` contenha `text/plain`. Essa é, na verdade, verificação de erro. Não é uma verificação de segurança. O tipo MIME é suposto pelo navegador do usuário a partir da extensão do arquivo e então passado ao servidor. Se houvesse alguma vantagem em passar um falso, não seria difícil para um usuário malicioso fazê-lo.

Então, verificamos se o arquivo que estamos tentando abrir foi realmente carregado e não é um arquivo local como `/etc/passwd`. Retornaremos a isso em breve.

Se isso funcionar bem, então copiamos o arquivo para nosso diretório include. Utilizamos `/uploads/` nesse exemplo – ele está fora da árvore de documento Web e, portanto, é um bom lugar para colocar arquivos que devem ser incluídos em outra parte.

Então abrimos o arquivo, limpamos qualquer tag de HTML ou PHP que possa estar no arquivo utilizando a função `strip_tags( )` e escrevemos o arquivo outra vez. Por fim, exibimos o conteúdo do arquivo, e o usuário pode ver que seu arquivo carregou com sucesso.

Os resultados de uma (bem-sucedida) execução desse script são mostrados na Figura 18.2. Em setembro de 2000, foi anunciada uma façanha que poderia permitir que um cracker enganasse o script do upload de arquivo no processamento de um arquivo local como se ele tivesse sido carregado. Essa façanha foi documentada na lista de mala-direta BUGTRAQ. Você pode ler o aviso oficial de segurança em um dos muitos repositórios de arquivos BUGTRAQ, como http://lists. insecure.org/bugtraq/2000/Sep/0237.html.

Utilizamos as funções `is_uploaded_file( )` e `move_uploaded_file( )` para certificar de que o upload do arquivo que estamos processando foi realmente feito e não é um arquivo local como `/etc/passwd`. Essa função está disponível a partir da versão PHP 4.0.3 em diante. Se você estiver utilizando uma versão mais antiga do PHP, novamente fornecemos algum código de exemplo com funcionalidade equivalente (desativado com caracteres de comentário).

Figura 18.2 Depois que o arquivo é copiado e reformatado, o arquivo carregado é exibido como confirmação para o usuário de que o arquivo foi carregado com sucesso.

A menos que você escreva seu script de tratamento de upload cuidadosamente, um visitante malicioso poderia fornecer seu próprio nome de arquivo temporário e convencer seu script a tratar isso como o arquivo carregado. Como muitos scripts de upload de arquivo ecoam os dados de volta para o usuário ou armazenam em algum lugar em que eles possam ser carregados, isso poderia fazer com que as pessoas fossem capazes de acessar qualquer arquivo que o servidor Web pudesse ler. Isso poderia incluir arquivos sigilosos como `/etc/passwd` e o código-fonte do PHP incluindo suas senhas de banco de dados.

Problemas comuns

Há alguns itens para se ter em mente ao realizar carregamentos de arquivos.

- O exemplo anterior assume que usuários foram autenticados em outra parte. Você não deve permitir que ninguém carregue arquivos no seu site.
- Se você estiver permitindo que usuários não-confiáveis ou sem autenticação carreguem arquivos, é uma boa idéia ser muito paranóico a respeito do conteúdo deles. Com certeza, você não quer um script malicioso sendo carregado e executado. Você deve ter cuidado, não apenas com o tipo e conteúdo do arquivo em que estamos aqui, mas com o próprio nome de arquivo. É uma ótima idéia renomear arquivos carregados para algo que você saiba que é "seguro".
- Se você estiver utilizando uma máquina baseada no Windows, certifique-se de utilizar \\ ou /, em vez de \ nos caminhos de arquivo como de costume.
- Utilizar o nome de arquivo fornecido pelo usuário como fizemos nessse script pode gerar uma série de problemas. O mais óbvio é que você corre o risco de sobrescrever acidentalmente arquivos existentes se alguém fizer upload de um arquivo com um nome que já foi usado. Um risco menos óbvio é que os diferentes sistemas operacionais e até diferentes configurações de linguagens locais permitem diferentes conjuntos de caracteres legais em nomes de arquivos. Um arquivo sendo carregado por upload pode ter um nome de arquivo com caracteres ilegais ao seu sistema.
- Se tiver problemas em fazer com que seu arquivo carregado funcione, verifique o arquivo `php.ini`. É preciso ter a diretiva `upload_tmp_dir` definida para apontar para algum diretório ao qual você tenha acesso. Também pode ser preciso ajustar a diretiva `memory_limit` se quiser carregar arquivos grandes; isso determina o tamanho máximo do arquivo em bytes que você pode carregar. O Apache também possui alguns tempos e limites de tamanho de transação configuráveis que merecem atenção se você tiver dificuldades com uploads grandes.

Utilizando funções de diretório

Depois que os usuários carregaram alguns arquivos, será útil a eles serem capazes de ver o que foi carregado e manipular os arquivos de conteúdo. O PHP tem um conjunto de funções de diretório e de sistema de arquivos que são úteis para esse propósito.

Lendo a partir de diretórios

Primeiro, implementaremos um script para permitir navegação pelo diretório do conteúdo carregado. Navegar diretórios é realmente muito simples e direto no PHP. Na Listagem 18.3, mostramos um script simples que pode ser utilizado para esse propósito.

Listagem 18.3 **browsedir.php – Uma listagem de diretório dos arquivos carregados**

```
<html>
<head>
  <title>Browse Directories</title>
</head>
<body>
<h1>Browsing</h1>
<?php
  $current_dir = '/uploads/';
  $dir = opendir($current_dir);

  echo "Upload directory is $current_dir<br />";
  echo 'Directory Listing:<br /><hr /><br />';
  while ($file = readdir($dir))
  {
      echo "$file<br />";
  }
  echo '<hr /><br />';
  closedir($dir);
?>
</body>
</html>
```

Esse script utiliza as funções `opendir( )`, `closedir( )` e `readdir( )`.

A função `opendir( )` é utilizada para abrir um diretório para leitura. Sua utilização é muito semelhante à utilização de `fopen( )` para ler a partir de arquivos. Em vez de passar a ele um nome de arquivo, você deve passar um nome de diretório:

```
$dir = opendir($current_dir);
```

A função retorna um handle de diretório, novamente quase da mesma maneira como `fopen( )` retorna um handle de arquivo.

Quando o diretório estiver aberto, você pode ler um nome de arquivo a partir dele chamando `readdir($dir)`, como mostrado no exemplo. Isso retorna `false` quando não houver mais arquivos para serem lidos. (Note que a função também retornará `false`se ler um arquivo chamado `"0"` – é claro que você poderia testar isso se houvesse alguma possibilidade de isso ocorrer.) Os arquivos não são classificados em nenhuma ordem particular; então, se solicitar uma lista classificada, você deve ler esses arquivos em um array e classificá-los antes de exibi-los.

Quanto tiver concluído a leitura de um diretório, você chama `closedir($dir)` para concluir. Isso é novamente semelhante à chamada `fclose( )` para um arquivo.

A saída de exemplo do script de navegação do diretório é mostrada na Figura 18.3.

Figura 18.3 A listagem de diretório mostra todos os arquivos no diretório escolhido, incluindo os diretórios . (o diretório atual) e .. (um nível acima). Você pode escolher filtrar esses.

Se você estiver tornando a navegação de diretório disponível via esse mecanismo, é sensato limitar os diretórios que podem ser navegados de modo que um usuário não possa navegar pelas listagens de diretório em áreas que não são normalmente disponíveis para ele.

Uma função associada e às vezes útil é `rewinddir($dir)`, que redefine a leitura de nomes de arquivo para o começo do diretório.

Como uma alternativa a essas funções, você pode utilizar a classe `dir` fornecida pelo PHP. Ela tem as propriedades `handle` e `path` e os métodos `read( )`, `close( )` e `rewind( )`, que funcionam identicamente às alternativas não-classe.

Obtendo informações sobre o diretório atual

Dado um caminho para um arquivo, podemos obter algumas informações adicionais.

As funções `dirname($caminho)` e `basename($caminho)` retornam a parte de diretório do caminho e a parte de nome de arquivo do caminho, respectivamente. Isso poderia ser útil para nosso navegador de diretório, particularmente se começarmos a acumular uma complexa estrutura de diretórios de conteúdo baseado em nomes de diretório significativos e nomes de arquivo.

Poderíamos também adicionar à nossa listagem de diretório uma indicação de quanto espaço é deixado para carregamentos utilizando a função `disk_free_space($caminho)`. Se você passar para essa função um caminho para um diretório, ela retornará o número de bytes livres no disco (Windows) ou no sistema de arquivos (UNIX) em que está o diretório.

Criando e excluindo diretórios

Além de ler passivamente informações sobre diretórios, você pode utilizar as funções de PHP `mkdir( )` e `rmdir( )` para criar e excluir diretórios. Você só será capaz de criar ou excluir diretórios nos caminhos a que o usuário em que o script é executado tem acesso.

Utilizar `mkdir( )` é mais complicado do que você poderia imaginar. Ele aceita dois parâmetros, o caminho para o diretório desejado (incluindo o novo nome de diretório) e as permissões que você gostaria que o diretório tivesse, por exemplo:

```
mkdir("/tmp/testing", 0777);
```

Entretanto, as permissões que você lista não são necessariamente as que obterá. O inverso do umask atual será combinado com esse valor usando `AND` para obter as permissões reais. Por exemplo, se o umask for 022, você obterá permissões de 0755.

Você poderia querer redefinir o umask antes de criar um diretório para evitar esse efeito, inserindo:

```
$oldumask = umask(0);
mkdir("/tmp/testing", 0777);
umask($oldumask);
```

Esse código utiliza a função `umask( )`, que pode ser utilizada para verificar e alterar o umask atual. Isso alterará o umask atual para o que for passado e retornará o umask antigo ou, se chamado sem parâmetros, apenas retornará o umask atual.

Observe que a função `umask( )` não tem nenhum efeito nos sistemas Windows.

A função `rmdir( )` exclui um diretório, da seguinte maneira:

```
rmdir("/tmp/testing");
```

ou:

```
rmdir("c:\\tmp\\testing");
```

O diretório que você está tentando excluir deve estar vazio.

Interagindo com o sistema de arquivos

Além de visualizar e obter informações sobre diretórios, podemos interagir com e obter as informações sobre arquivos no servidor Web. Vimos anteriormente gravação e leitura de arquivos. Uma grande quantidade de outras funções de arquivo está disponível.

Obtendo informações de arquivo

Podemos alterar parte de nosso script de navegação de diretório para ler arquivos da seguinte maneira:

```
while ($file = $dir->read( ))
{
 echo '<a href="filedetails.php?file='.$file.'">'.$file.'</a><br />';
}
```

Podemos então criar o script `filedetails.php` para fornecer informações adicionais sobre um arquivo. O conteúdo desse arquivo é mostrado na Listagem 18.4.

Um aviso sobre esse script: algumas funções utilizadas aqui não são suportadas sob o Windows, incluindo `posix_getpwuid( )` e `fileowner( )`, ou não são suportadas confiavelmente.

Listagem 18.4 **`filedetails.php` – Funções de status de arquivo e seus resultados**

```
<html>
<head>
  <title>File Details</title>
</head>
<body>
<?php
  $current_dir = '/uploads/';
  $file = basename($file);  // elimina informações de diretório para segurança

  echo '<h1>Details of file: '.$file.'</h1>';
  $file = $current_dir.$file;

  echo '<h2>File data</h2>';
  echo 'File last accessed: '.date('j F Y H:i', fileatime($file)).'<br />';
  echo 'File last modified: '.date('j F Y H:i', filemtime($file)).'<br />';

  $user = posix_getpwuid(fileowner($file));
  echo 'File owner: '.$user['name'].'<br />';
```

Listagem 18.4 Continuação

```
  $group = posix_getgrgid(filegroup($file));
  echo 'File group: '.$group['name'].'<br />';

  echo 'File permissions: '.decoct(fileperms($file)).'<br />';

  echo 'File type: '.filetype($file).'<br />';

  echo 'File size: '.filesize($file).' bytes<br />';


  echo '<h2>File tests</h2>';

  echo 'is_dir: '.(is_dir($file)? 'true' : 'false').'<br />';
  echo 'is_executable: '.(is_executable($file)? 'true' : 'false').'<br />';
  echo 'is_file: '.(is_file($file)? 'true' : 'false').'<br />';
  echo 'is_link: '.(is_link($file)? 'true' : 'false').'<br />';
  echo 'is_readable: '.(is_readable($file)? 'true' : 'false').'<br />';
  echo 'is_writable: '.(is_writable($file)? 'true' : 'false').'<br />';

?>
</body>
</html>
```

Os resultados de uma execução de exemplo da Listagem 18.4 são mostrados na Figura 18.4.

Vamos conversar sobre o que faz cada uma das funções utilizadas na Listagem 18.4. Como mencionado anteriormente, a função `basename( )` obtém o nome do arquivo sem o diretório. (Você também pode utilizar a função `dirname( )` para obter o nome de diretório sem o nome de arquivo.)

As funções `fileatime( )` e `filemtime( )` retornam o registro de data e hora em que o arquivo foi acessado e modificado pela última vez, respectivamente. Reformatamos o registro de data e hora utilizando a função `date( )` para torná-lo mais legível por humanos. Essas funções retornarão o mesmo valor em alguns sistemas operacionais (como no exemplo) dependendo de quais informações o sistema armazena.

As funções `fileowner( )` e `filegroup( )` retornam o ID de usuário (`uid`) e o ID de grupo (`gid`) do arquivo. Esses podem ser convertidos para nomes utilizando as funções `posix_getpwuid( )` e `posix_getgrgid( )`, respectivamente, que os tornam mais legíveis. Essas funções aceitam o `uid` ou o `gid` como um parâmetro e retornam um array associativo das informações sobre o usuário ou grupo, incluindo o nome do usuário ou grupo, como utilizamos nesse script.

A função `fileperms( )` retorna as permissões no arquivo. Nós as reformatamos como um número octal utilizando a função `decoct( )` para colocá-las em um formato mais familiar para usuários de UNIX.

A função `filetype( )` retorna algumas informações sobre o tipo de arquivo sendo examinado. Os possíveis resultados são `fifo`, `char`, `dir`, `block`, `link`, `file` e `unknown`.

A função `filesize( )` retorna o tamanho do arquivo em bytes.

O segundo conjunto de funções – `is_dir( )`, `is_executable(<|>)`, `is_file( )`, `is_link( )`, `is_readable( )` e `is_writable( )` – inteiro testa o atributo nomeado de um arquivo e retorna true ou false.

Alternativamente, poderíamos ter utilizado a função `stat( )` para reunir uma grande quantidade das mesmas informações. Quando a função recebe um arquivo, esse retorna um array contendo dados semelhantes a essas funções. A função `lstat( )` é semelhante, mas para utilização com links simbólicos.

Todas as funções de status de arquivo são bem caras para executar em termos de tempo. Seus resultados são, portanto, armazenados em cache. Se quiser verificar algumas informações de arquivo antes e depois de uma alteração, você precisa chamar:

Figura 18.4 A visualização File Details mostra informações de sistema de arquivos sobre um arquivo. Observe que permissões são mostradas em um formato octal.

```
clearstatcache( );
```

para limpar os resultados anteriores. Se quiser utilizar o script anterior antes e depois de alterar alguns dos dados de arquivo, você deve começar chamando essa função para certificar-se de que os dados produzidos são atuais.

Alterando propriedades de arquivo

Além de visualizar propriedades de arquivo, você pode alterá-las.

Todas as funções `chgrp(`*`arquivo, grupo`*`)`, `chmod(`*`arquivo, permissões`*`)` e `chown(`*`arquivo, usuário`*`)` comportam-se de maneira semelhante a seu equivalente UNIX. Nenhuma dessas funcionará em sistemas baseados no Windows, embora `chown( )` executará e sempre retornará `true`.

A função `chgrp( )` é utilizada para alterar o grupo de um arquivo. Ela só pode ser utilizada a fim de alterar para grupos dos quais o usuário seja membro, a menos que o usuário seja root.

A função `chmod( )` é utilizada para alterar as permissões em um arquivo. As permissões que você passa a ela estão na forma `chmod` UNIX normal – você deve prefixá-las com um 0 (zero) para mostrar que eles estão em octal, por exemplo:

```
chmod('somefile.txt', 0777);
```

A função `chown( )` é utilizada para alterar o proprietário de um arquivo. Ela só pode ser utilizada se o script estiver executando como root, o que não deve nunca acontecer.

Criando, excluindo e movendo arquivos

Você pode utilizar funções do sistema de arquivos para criar, mover e excluir arquivos.

Primeiro e bem simplesmente, você pode criar um arquivo ou alterar a data/hora da última modificação, utilizando a função `touch( )`. Ela funciona de maneira semelhante ao comando `touch` do UNIX. A função tem o seguinte protótipo:

```
int touch (string arquivo, [int data/hora [, int data/hora_alternativa]])
```

Se o arquivo já existir, a data/hora da modificação será alterada para a data/hora atual ou a data/hora dada no segundo parâmetro, se for especificado. Se quiser especificar isso, você deve fazê-lo

no formato de registro de data e hora. Se o arquivo não existir, ele será criado. A data/hora de acesso do arquivo também mudará: por padrão para a data/hora atual do sistema ou, alternativamente, para o registro de data/hora que você especifica no parâmetro *data/hora_alternativa opcional.*

Você pode excluir arquivos utilizando a função `unlink( )`. (Note que essa função não é chamada `delete` – nem há nenhuma exclusão.) Você a utiliza assim:

```
unlink($filename);
```

Essa é uma das funções que não funciona com algumas versões mais antigas do Windows. Entretanto, se não funcionar na sua configuração, você poderá excluir um arquivo no Windows com

```
system("del filename.ext");
```

Você pode copiar e mover arquivos com as funções `copy( )` e `rename( )`, da seguinte maneira:

```
copy($source_path, $destination_path);
rename($oldfile, $newfile);
```

Talvez você tenha notado que utilizamos `copy( )` na Listagem 18.2.

A função `rename( )` faz duas tarefas diferentes como uma função para mover arquivos de um lugar para outro porque o PHP não tem uma função `move`. A possibilidade de mover arquivos de um sistema de arquivos para outro, e de os arquivos serem sobrescritos quando `rename( )` é utilizada depende do sistema operacional, então verifique os efeitos em seu servidor. Além disso, tenha cuidado com o caminho que você utiliza para o nome de arquivo. Se relativo, esse será em relação à localização do script, não ao arquivo original.

Utilizando funções de execução de programa

Mudaremos agora das funções de sistema de arquivos e examinaremos as funções que estão disponíveis para executar comandos no servidor.

Isso é útil quando você deve fornecer um front end baseado na Web para um sistema existente baseado em linha de comando. Por exemplo, utilizamos esses comandos para configurar um front end para o gerenciador de listas de mala-direta `ezmlm`. Utilizaremos estes novamente quando chegarmos aos estudos de caso mais adiante neste livro.

Há quatro técnicas principais que você pode utilizar para executar um comando no servidor Web. Essas técnicas são todas muitos semelhantes, mas há algumas diferenças menores.

1. `exec( )` – A função `exec( )` tem o seguinte protótipo:

   `string exec (string` *`comando`* `[, array` *`resultado`* `[, int` *`valor_de_retorno`*`]])`

   Você passa o comando que gostaria de executar, por exemplo:

   ```
   exec("ls -la");
   ```

   A função `exec( )` não tem saída direta.

   Essa função retorna a última linha do resultado do comando.

   Se passar uma variável como *`resultado`*, você obterá de volta um array de strings representando cada linha da saída. Se passar uma variável como *`valor_retorno`*, você obterá o código de retorno.

2. `passthru( )` - A função `passthru( )` tem o seguinte protótipo:

   `void passthru (string` *`comando`* `[, int` *`valor_de_retorno`*`])`

   A função `passthru( )` ecoa diretamente a saída do navegador. (Isso é útil se a saída for binária, por exemplo, algum tipo de dados de imagem.) Ela não retorna nada.

Os parâmetros funcionam da mesma maneira que os parâmetros `exec( )`.

3. `system( )` – A função `system( )` tem o seguinte protótipo:

   `string system (string comando [, int valor_de_retorno])`

   A função ecoa a saída do comando para o navegador. Essa função tenta limpar a saída depois de cada linha (assumindo que você está executando o PHP como um módulo de servidor), que a distingue de `passthru( )`. A função `system( )` retorna a última linha da saída (em caso de sucesso) ou `false` (em caso de falha).

   Os parâmetros funcionam da mesma maneira que as outras funções.

4. Sinais de crase – Mencionamos isso brevemente no Capítulo 1. Esses são realmente um operador de execução.

   Eles não têm saída direta. O resultado de executar o comando é retornado como uma string, que então pode ser ecoada ou o que você quiser.

Se você tiver necessidades mais complexas, também poderá utilizar `popen( )`, `proc_open( )` e `proc_close( )`, que são utilizadas para bifurcar processos externos e para redirecionar dados entre eles. As duas últimas funções foram adicionadas ao PHP 4.3.

O script mostrado na Listagem 18.5 ilustra como utilizar cada uma dessas quatro técnicas de modo equivalente.

Listagem 18.5 **progex.php – Funções de status de arquivo e seus resultados**

```
<?php

   chdir('/uploads/');

///// versão exec
   echo '<pre>';

   // unix
   exec('ls -la', $result);
   // Windows
   // exec ('dir', $result);
   foreach ($result as $line)
     echo "$line\n";

   echo '</pre>';
   echo '<br /><hr /><br />';

///// versão passthru
   echo '<pre>';

   // unix
   passthru('ls -la');
   // Windows
   // passthru ('dir');

   echo '</pre>';
   echo '<br /><hr /><br />';

///// versão de sistema

   echo '<pre>';
   // unix
   $result = system('ls -la');
   // Windows
```

Listagem 18.5 **Continuação**

```
    // $result = system ('dir');
    echo '</pre>';
    echo '<br /><hr /><br />';

/////versão com sinais de crase
   echo '<pre>';
    // unix
    $result = 'ls -al';
    // Windows
    // $result = "dir";
    echo $result;
    echo '</pre>';

?>
```

Poderíamos ter utilizado uma dessas abordagens como uma alternativa para o script de navegação de diretório que escrevemos anteriormente. Observe que um dos efeitos colaterais da utilização de funções externas é amplamente demonstrado por esse código – seu código não é mais portável. Utilizamos comandos Unix aqui e, claramente, o código não será executado no Windows.

Se planeja incluir dados enviados pelo usuário como parte do comando que executará, você sempre deve executá-lo primeiro pela função `escapeshellcmd( )`. Isso impede que os usuários executem maliciosamente (ou de outro modo) comandos no sistema. Você pode chamá-la assim, por exemplo:

```
system(escapeshellcmd($command));
```

Você também deve utilizar a função `escapeshellarg( )` para "escapar" quaisquer argumentos que planeja passar para seu comando de shell.

Interagindo com o ambiente: getenv( ) e putenv( )

Antes de deixarmos esta seção, veremos como você pode utilizar as variáveis de ambiente a partir de dentro do PHP. Há duas funções para esse propósito: `getenv( )`, que permite recuperar variáveis de ambiente, e `putenv( )`, que permite configurar variáveis de ambiente.

Observe que o ambiente que estamos discutindo aqui é o ambiente em que o PHP executa no servidor.

Você pode obter uma lista de todas as variáveis de ambiente PHP executando `phpinfo( )`. Algumas são mais úteis que outras; por exemplo,

```
getenv("HTTP_REFERER");
```

retornará o URL da página que levou o usuário para a página atual.

Você também pode configurar as variáveis de ambiente conforme necessário com `putenv( )`; por exemplo:

```
$home = "/home/nobody";
putenv (" HOME=$home ");
```

Se você for um administrador de sistema e quiser limitar quais variáveis de ambiente os programadores podem configurar, poderá utilizar a diretiva `safe_mode_allowed_env_vars` no `php.ini`. Quando o PHP é executado no modo de segurança, os usuários somente serão capazes de configurar variáveis de ambiente cujos prefixos estejam listados nessa diretiva.

> Se quiser informações adicionais sobre o que representam algumas variáveis de ambiente, você pode ver a especificação CGI:
> http://hoohoo.ncsa.uiuc.edu/cgi/env.html

Leitura adicional

A maioria das funções de sistema de arquivos no PHP mapeia funções subjacentes do sistema operacional – experimente ler as páginas man se estiver utilizando UNIX para obter informações adicionais.

A seguir

No Capítulo 19, utilizaremos as funções de protocolo e rede do PHP para interagir com outros sistemas diferentes do nosso próprio servidor Web. Isso novamente expande os horizontes do que podemos fazer com nossos scripts.

19

Utilizando funções de rede e protocolo

NESTE CAPÍTULO, VEREMOS AS FUNÇÕES BASEADAS em rede do PHP que permitem que scripts interajam com o resto da Internet. Há um mundo de recursos por aí afora e uma grande variedade de protocolos disponíveis para utilizá-los.

Os tópicos-chave deste capítulo incluem:

- Examinando protocolos disponíveis;
- Enviando e lendo e-mail;
- Utilizando outros Web sites via http;
- Utilizando funções de pesquisa de rede;
- Utilizando FTP.

Examinado protocolos disponíveis

Os protocolos são as regras de comunicação para uma determinada situação. Por exemplo, você conhece o protocolo ao encontrar outra pessoa: você diz oi, troca um aperto de mãos, comunica-se por algum tempo e depois se despede. Os protocolos de rede de computador são semelhantes.

Como os protocolos humanos, os diferentes protocolos de computador são utilizados para diferentes situações e aplicações. Utilizamos o HTTP, o Hypertext Transfer Protocol, para enviar e receber páginas Web. Você provavelmente também já utilizou o FTP, File Transfer Protocol, para transferir arquivos entre máquinas em uma rede. Há muitos outros.

Os protocolos e outros padrões da Internet (Internet Standards), são descritos em documentos chamados RFCs ou *Requests for Comments*. Esses protocolos são definidos pelo Internet Engineering Task Force (IETF). As RFCs estão amplamente disponíveis na Internet. A fonte básica é o RFC Editor em:

http://www.rfc-editor.org/

Se você tiver problemas ao trabalhar com um dado protocolo, as RFCs são a fonte autorizada e são freqüentemente úteis para solucionar problemas com seu código. As RFCs são, porém, muito detalhadas e freqüentemente ocupam centenas de páginas.

Alguns exemplos de RFCs famosas são a RFC2616, que descreve o protocolo HTTP/1.1, e a RFC822, que descreve o formato de mensagens de e-mail da Internet.

Neste capítulo, examinaremos aspectos do PHP que utilizam alguns desses protocolos. Especificamente, falaremos sobre como enviar correio com SMTP, ler correio com o POP e o IMAP, conectar-se a outros servidores Web via HTTP e HTTPS e transferir arquivos com o FTP.

Enviando e lendo e-mail

A principal maneira de enviar correio no PHP é utilizar a simples função `mail( )`. Discutimos o uso dessa função no Capítulo 4, então não a visitaremos novamente aqui. Essa função utiliza SMTP (Simple Mail Transfer Protocol) para enviar correio.

Você pode utilizar uma variedade de classes livremente disponíveis para adicionar a funcionalidade de `mail( )`. No Capítulo 30, utilizaremos uma classe suplementar para enviar anexos de HTML com uma parte do correio. O SMTP serve somente para enviar correio. Os protocolos IMAP (Internet Message Access Protocol, descritos na RFC2060) e o POP (Post Office Protocol, descrito na RFC1939 ou STD0053) são utilizados para ler correio proveniente de um servidor de correio. Esses protocolos não podem enviar correio.

O IMAP é utilizado para ler e manipular mensagens de correio armazenadas em um servidor e é mais sofisticado que o POP, que em geral é utilizado simplesmente para fazer download de mensagens de correio para um cliente e excluí-las do servidor.

O PHP vem com uma biblioteca de IMAP. Essa também pode ser utilizada para fazer conexões POP e NNTP (Network News Transfer Protocol) bem como conexões IMAP.

Examinaremos extensamente o uso da biblioteca de IMAP no projeto descrito no Capítulo 29.

Utilizando outros Web sites

Uma das excelentes ações possíveis com a Web é utilizar, modificar e embutir serviços e informações existentes nas suas próprias páginas. O PHP torna isso muito fácil. Vejamos um exemplo para ilustrar isso.

Imagine que a empresa em que você trabalha quisesse a cotação das ações de sua empresa exibida em sua home page. Essas informações estão disponíveis em algum site de bolsa de valores em algum lugar – mas como as obtemos?

Comece localizando um URL da fonte original das informações. Quando descobrir isso, toda vez que alguém visitar sua home page, você pode abrir uma conexão com esse URL, recuperar a página e obter as informações necessárias.

Como um exemplo, montamos um script que recupera e reformata uma cotação de ação do Web site da AMEX. Para o propósito do exemplo, recuperamos o preço atual das cotações de ações da Amazon.com. (As informações que você quer incluir na sua página provavelmente serão diferentes, mas os princípios são os mesmos.)

Essa técnica é conhecida como *sreen scraping* porque você pega as informações que deveriam ser exibidas em uma tela e extrai as partes necessárias para apresentação com uma nova interface. Esse script é mostrado na Listagem 19.1.

Listagem 19.1 **`lookup.php` – Script que recupera cotações da NASDAQ para a ação com o símbolo listado em `$symbol`**

```
<html>
<head>
  <title>Stock Quote from NASDAQ</title>
</head>
<body>
<?php
  // escolhe uma ação da bolsa para ver
  $symbol='AMZN';
  echo "<h1>Stock Quote for $symbol</h1>";

  $theurl='http://www.amex.com/equities/listCmp/EqLCDetQuote.jsp?Product_Symbol=AMZN';
 if (!($fp = fopen($theurl, 'r')))
  {
    echo 'Could not open URL';
```

Listagem 19.1 Continuação

```
    exit;
  }
  $contents = fread($fp, 1000000);
  fclose($fp);

  //echo $contents;

  // localiza a parte da página que desejamos e gera sua saída
  $pattern = "(\\\$[0-9 ]+\\.[0-9]+)";
  if (eregi($pattern, $contents, $quote))
  {
    echo "$symbol was last sold at: ";
    echo $quote[1];
  } else
  {
    echo 'No quote available';
  };

  // reconhece a fonte
  echo '<br />'
      .'This information retrieved from <br />'
      ."<a href=\"$theurl\">$theurl</a><br />"
      .'on '.(date('l jS F Y g:i a T'));
?>
</body>
</html>
```

A saída de uma execução de exemplo da Listagem 19.1 é mostrada na Figura 19.1.

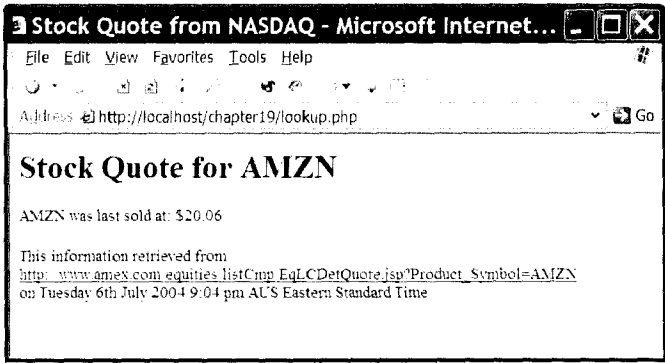



Figura 19.1 O script utiliza uma expressão regular para extrair a cotação de uma ação a partir das informações recuperadas da bolsa de valores.

O script em si é muito simples e direto – de fato, ele não utiliza nenhuma função que não tenhamos visto antes, apenas novas aplicações dessas funções.

Talvez você se lembre de que quando discutimos a leitura de arquivos no Capítulo 2, mencionamos que você poderia utilizar as funções de arquivo para ler a partir de um URL. Foi isso que fizemos nesse caso. A chamada a `file_get_contents( )`

```
$contents = file_get_contents($theurl)
```

retorna o texto completo da página Web que o URL armazenado em `$contents`.

As funções de arquivo podem fazer muito no PHP. O exemplo aqui simplesmente carrega uma página Web via HTTP, mas você poderia interagir com outros servidores via HTTPS, FTP ou outros protocolos exatamente da mesma forma. Para algumas tarefas, você pode precisar de uma

abordagem mais especializada. Alguma funcionalidade FTP está disponível nas funções específicas FTP, e não via fopen( ) e outras funções de arquivo. Temos um exemplo que utiliza funções FTP posteriormente neste capítulo. Para algumas tarefas HTTP ou HTTPS, você pode precisar utilizar a biblioteca cURL. Com cURL, é possível fazer log in em um Web site e imitar o progresso de um usuário por algumas páginas.

Depois de obter o texto da página de file_get_contents( ), você pode utilizar uma expressão regular e a função eregi( ) para encontrar a página desejada:

```
$pattern = "(\\\$[0-9 ]+\\.[0-9]+)";
if (eregi($pattern, $contents, $quote))
{
  echo "$symbol was last sold at: ";
  echo $quote[1];
}
```

É isso aí!

Você pode utilizar essa abordagem para uma variedade de propósitos. Outro bom exemplo é recuperar informações locais sobre o clima e incorporá-las na sua página.

A melhor utilização dessa abordagem é combinar informações de diferentes fontes para adicionar algum valor. Um bom exemplo dessa abordagem pode ser visto no script infame do Philip Greenspun que produz o Bill Gates Wealth Clock:

http://www.webho.com/WealthClock

Essa página extrai informações de duas fontes. Ela obtém a população atual dos Estados Unidos a partir do site do U.S. Census Bureau. Essa página pesquisa o valor atual das ações da Microsoft e combina essas duas partes das informações, adiciona uma dose saudável da opinião do autor e produz novas informações – uma estimativa da fortuna atual de Bill Gates.

Uma nota paralela: se você estiver utilizando fonte externa de informações como essa para um propósito comercial, é uma boa idéia consultar a fonte primeiro. Há questões de propriedade intelectual a serem consideradas em alguns casos.

Se estiver construindo um script assim, talvez você queira passar alguns dados. Por exemplo, se você se conectar a um URL externo, talvez queira passar alguns parâmetros digitados pelo usuário. Se for fazer isso, é uma boa idéia utilizar a função urlencode( ). Isso aceitará uma string e a converterá no formato adequado para um URL, por exemplo, transformando espaços em sinais de adição. Você pode chamá-la assim:

```
$encodedparameter = urlencode($parameter);
```

Um problema com essa abordagem geral é que o site do qual você está obtendo as informações pode alterar seu formato de dados, o que impedirá o script de funcionar.

Uma forma melhor de fazer a mesma coisa é usar Web services. Esses serviços são como objetos remotos que você pode conectar para recuperar dados como cotação da Bolsa. O suporte PHP para Web Services está crescendo. Além das vantagens técnicas, utilizar Web Services significa que você está usando uma interface e uma facilidade de que provedor forneceu implícita ou explicitamente permissão a você de acessá-lo com um programa.

Utilizando funções de pesquisa da rede

O PHP oferece um conjunto de funções de "pesquisa" que pode ser utilizado para verificar as informações sobre nome de hosts, endereços IP e trocas de correio. Por exemplo, se você estivesse configurando um site de diretório como o Yahoo! quando novos URLs fossem enviados, talvez quisesse automaticamente verificar se o host de um URL e as informações de contato para esse site são válidos. Dessa maneira, você pode evitar alguns custos adicionais durante o trabalho quando um revisor examinar um site e descobrir que ele não existe ou que o endereço de e-mail não é válido.

A Listagem 19.2 mostra a HTML para um formulário de envio para um diretório assim.

Listagem 19.2 **directory_submit.html** – HTML para o formulário de envio

```
<head>
  <title>Submit your site</title>
</head>
<body>
<h1>Submit site</h1>
<form method="post" action="directory_submit.php">
URL: <input type="text" name="url" size="30" value="http://"><br />
Email contact: <input type="text" name="email" size="23"><br />
<input type="submit" name="Submit site">
</form>
</body>
</html>
```

Esse é um formulário muito simples – a versão gerada no navegador, com alguns dados inseridos de exemplo, é mostrada na Figura 19.2.

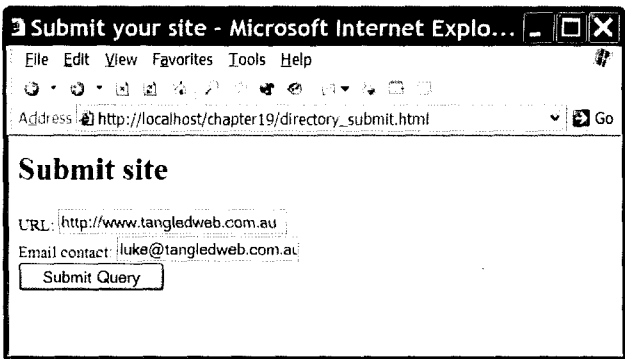



Figura 19.2 As submissões de diretório em geral requerem seu URL e alguns detalhes de contato de modo que administradores do diretório possam notificá-lo quando seu site for adicionado ao diretório.

Quando o botão submit for pressionado, queremos verificar, primeiro, se o URL está hospedado em uma máquina real, e, segundo, se a parte host do endereço de e-mail também está em uma máquina real. Escrevemos um script para verificar isso e a saída é mostrada na Figura 19.3.

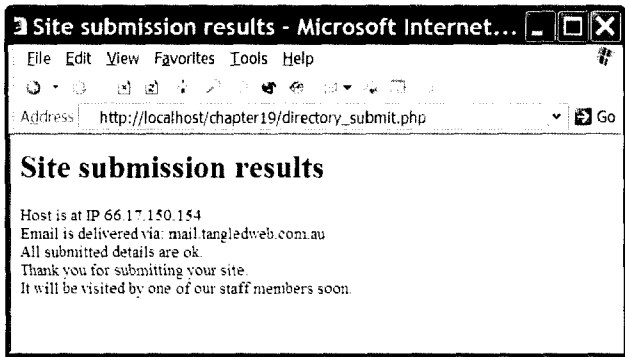



Figura 19.3 Essa versão do script exibe os resultados da verificação de nomes de host do URL e o endereço de e-mail – uma versão de produção talvez não exiba esses resultados, mas é interessante ver as informações retornadas de nossa verificação.

O script que realiza essas verificações utiliza duas funções provenientes do conjunto de funções de rede do PHP – `gethostbyname( )` e `getmxrr( )`. O script completo é mostrado na Listagem 19.3.

Listagem 19.3 **directory_submit.php** – Script para verificar URL e endereço de e-mail

```
<html>
<head>
  <title>Site submission results</title>
</head>
<body>
<h1>Site submission results</h1>
<?php

  // Extrai campos de formulário

  $url = $HTTP_POST_VARS['url'];
  $email = $HTTP_POST_VARS['email'];

  // Verifica o URL

  $url = parse_url($url);
  $host = $url['host'];
  if(!($ip = gethostbyname($host)))
  {
    echo 'Host for URL does not have valid IP';
    exit;
  }

  echo "Host is at IP $ip <br />";

  // Verifica o endereço de e-mail

  $email = explode('@', $email);
  $emailhost = $email[1];

  // note que a função getmxrr( ) *não é implementada* em
  // versões Windows do PHP
  if (!getmxrr($emailhost, $mxhostsarr))
  {
    echo 'Email address is not at valid host';
    exit;
  }

  echo 'Email is delivered via: ';
  foreach ($mxhostsarr as $mx)
    echo "$mx ";

  // Se chegou até aqui, está tudo ok

  echo '<br />All submitted details are ok.<br />';
  echo 'Thank you for submitting your site.<br />'
       .'It will be visited by one of our staff members soon.'

  // Em um caso real, adiciona ao db de sites em espera...
?>
</body>
</html>
```

Vamos examinar individualmente as partes interessantes desse script.

Primeiro, pegamos o URL e aplicamos a função parse_url( ) a ele. Essa função retorna um array associativo das diferentes partes de um URL. As partes disponíveis de informações são: scheme, user, pass, host, port, path, query e fragment. Em geral, você não vai precisar de todas essas partes, mas eis um exemplo de como elas compõem um URL.

Dado um URL como:

http://nobody:secret@bigcompany.com:80/script.php?variable=value#anchor

Os valores de cada uma das partes do array seriam:

- scheme: http://
- user: nobody
- pass: secret
- host: bigcompany.com
- port: 80
- path: script.php
- query: variable=value
- fragment: anchor

No script `directory_submit`, queremos apenas as informações host, então as extraímos do array da seguinte maneira:

```
$url = parse_url($url);
$host = $url['host'];
```

Depois de fazer isso, podemos obter o endereço de IP desse host, se estiver no DNS. Podemos fazer isso utilizando a função `gethostbyname( )`, que retornará o IP se houver algum, ou `false` se não:

```
$ip = gethostbyname($host);
```

Você também pode fazer de outra maneira utilizando a função `gethostbyaddr( )`, que recebe um IP como parâmetro e retorna o nome do host. Se chamar essas funções em sucessão, você também pode terminar com um nome de host diferente daquele com que iniciou. Isso pode significar que um site está utilizando um serviço de hospedagem virtual.

Se o URL for válido, então continuamos a verificar o endereço de e-mail. Primeiro, nós o dividimos em nome de usuário e nome de host com uma chamada a `explode( )`:

```
$email = explode('@', $email);
$emailhost = $email[1];
```

Quando tivermos a parte host do endereço, podemos verificar se há um lugar para esse correio utilizar a função `dns_get_mx( )`:

```
dns_get_mx($emailhost, $mxhostsarr);
```

Essa função retorna o conjunto dos registros MX (Mail Exchange) de um endereço no array que você fornece em `$mxhostarr`.

Um registro MX é armazenado no DNS e é pesquisado como um hostname. A máquina listada no registro de MX não é necessariamente a máquina em que o e-mail por fim termina. Em vez disso, é uma máquina que sabe para onde rotear esse e-mail. (Há mais de uma; portanto, essa função retorna um array em vez de uma string de nomes de host.) Se não tivermos um registro MX no DNS, não há nenhum lugar para o correio ir.

Note que a função `dns_get_mx( )` não é implementada em versões Windows do PHP. Se estiver utilizando o Windows, examine o pacote PEAR::Net_DNS, que funcionará para você.

Se todas essas verificações estiverem certas, podemos colocar os dados desse formulário em um banco de dados para posterior revisão por um funcionário.

Além das funções que acabamos de utilizar, podemos utilizar a função mais genérica `checkdnsrr( )`, que aceita um nome de host e retorna `true` se houver algum registro dele no DNS.

Utilizando FTP

O File Transfer Protocol, ou FTP, é utilizado para transferir arquivos entre hosts em uma rede. Utilizando o PHP, você pode utilizar `fopen( )` e as várias funções de arquivo com o FTP da mesma maneira que faz com conexões de HTTP, para conectar-se a arquivos e transferir arquivos para e de um servidor FTP. Mas também há um conjunto de funções específicas do FTP que vem com a instalação de PHP padrão.

Essas funções não são predefinidas na instalação padrão por default. Para utilizá-las sob o UNIX, você precisará executar o programa `configure` do PHP com a opção `--enable-ftp` e então executar novamente `make`. Se você estiver usando a instalação do Windows padrão, as funções de FTP serão ativadas automaticamente.

(Para obter mais detalhes sobre a configuração do PHP, consulte o Apêndice A.)

Utilizando FTP para fazer backup ou espelhar um arquivo

As funções de FTP são úteis para mover e copiar arquivos de e para outros hosts. Uma utilização comum que você talvez faça disso é fazer backup do seu Web site ou espelhar arquivos em outra localização. Veremos um exemplo simples que utiliza as funções de FTP para espelhar um arquivo. Esse script é mostrado na Listagem 19.4.

Listagem 19.4 **`ftpmirror.php` – Script para fazer download das novas versões de um arquivo a partir de um servidor FTP**

```
<html>
<head>
  <title>Mirror update</title>
</head>
<body>
<h1>Mirror update</h1>
<?php
// configura variáveis - altera estas para adaptar o script à sua aplicação
$host = 'ftp.cs.rmit.edu.au';
$user = 'anonymous';
$password = 'laura@tangledweb.com.au';
$remotefile = '/pub/tsg/teraterm/ttssh14.zip';
$localfile = '/tmp/writable/ttssh14.zip';

// conecta-se ao host
$conn = ftp_connect("$host");
if (!$conn)
{
  echo 'Error: Could not connect to ftp server<br />';
  exit;
}
echo "Connected to $host.<br />";

// faz download do arquivo
@ $result = ftp_login($conn, $user, $pass);
if (!$result)
{
  echo "Error: Could not log on as $user<br />";
  ftp_quit($conn);
  exit;
}
echo "Logged in as $user<br />";
// verifica a data/hora do arquivo para ver se uma atualização é necessária
echo 'Checking file time...<br />';
```

Listagem 19.4 **Continuação**

```
if (file_exists($localfile))
{
  $localtime = filemtime($localfile);
  echo 'Local file last updated ';
  echo date('G:i j-M-Y', $localtime);
  echo '<br />';
}
else
  $localtime=0;
$remotetime = ftp_mdtm($conn, $remotefile);
if (!($remotetime >= 0))
{
  // Isso não significa que o arquivo não está lá, o servidor pode não suportar mod time
   echo 'Can\'t access remote file time.<br />';
   $remotetime=$localtime+1;  // certifica-se de uma atualização
}
else
{
  echo 'Remote file last updated ';
  echo date('G:i j-M-Y', $remotetime);
  echo '<br />';
}
if (!($remotetime > $localtime))
{
   echo 'Local copy is up to date.<br />';
   exit;
}

// faz download do arquivo
echo 'Getting file from server...<br />';
$fp = fopen ($localfile, 'w');
if (!$success = ftp_fget($conn, $fp, $remotefile, FTP_BINARY))
{
  echo 'Error: Could not download file';
  ftp_quit($conn);
  exit;
}
fclose($fp);
echo 'File downloaded successfully';

// fecha conexão com o host
ftp_quit($conn);

?>
</body>
</html>
```

A saída da execução desse script em uma ocasião é mostrada na Figura 19.4.

O script ftp_mirror.php é bem genérico. Você verá que ele começa configurando algumas variáveis:

```
$host = 'ftp.cs.rmit.edu.au';
$user = 'anonymous';
$password = 'laura@tangledweb.com.au';
$remotefile = '/pub/tsg/teraterm/ttssh14.zip';
$localfile = "/tmp/writable/ttssh14.zip";
```

A variável $host deve conter o nome do servidor FTP ao qual você deseja se conectar e $user e $password correspondentes ao nome de usuário e à senha com os quais você gostaria de efetuar o logon.

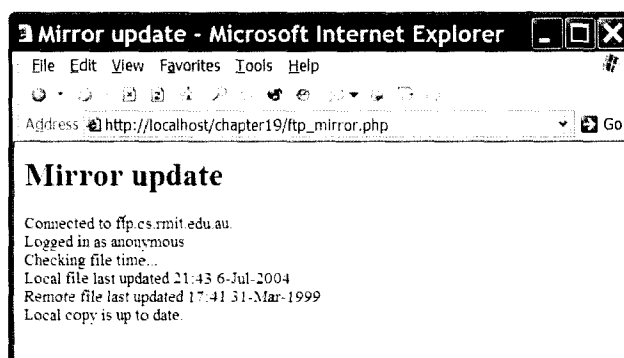



Figura 19.4 O script de espelhamento de FTP verifica se a versão local de um arquivo está atualizada, se não estiver, ele faz download de uma nova versão.

Muitos sites FTP suportam o chamado *login anônimo,* isto é, um nome de usuário livremente disponível que qualquer pessoa possa utilizar para se conectar. Não é necessária nenhuma senha, mas é uma cortesia comum fornecer seu endereço de e-mail como uma senha para que os administradores de sistema possam ver de onde vêm os usuários. Seguimos essa convenção aqui.

A variável `$remotefile` contém o caminho para o arquivo com que gostaríamos de fazer download. Nesse caso, estamos fazendo download e estamos espelhando uma cópia local de Tera Term SSH, um cliente de SSH para Windows. (SSH significa *secure shell.* Essa é uma forma encriptada de telnet.)

A variável `$localfile` contém o caminho para a localização em que armazenaremos o arquivo descarregado em nossa máquina. Nesse caso, criamos um diretório chamado `/tmp/writable` com permissões configuradas de modo que o PHP possa gravar um arquivo nele. Independente do sistema operacional, você precisa criar esse diretório para o script funcionar. Se seu sistema operacional possui permissões restritas, você precisará certificar-se de que permite que seu script seja gravado. Você deve ser capaz de alterar essas variáveis para adaptar esse script a seus propósitos.

Os passos básicos que seguimos nesse script são os mesmos que utilizaríamos se quiséssemos enviar o arquivo manualmente por FTP a partir de uma interface de linha de comando:

1. Conecte-se ao servidor FTP remoto.
2. Efetue logon (como um usuário ou anônimo).
3. Verifique se o arquivo remoto foi atualizado.
4. Se foi atualizado, faça download dele.
5. Feche a conexão de FTP.

Vamos examinar individualmente cada um desses.

Conectando-se ao servidor FTP remoto

O primeiro passo é equivalente a digitar

```
ftp hostname
```

em um prompt de comando em uma plataforma Windows ou UNIX. Realizamos esse passo no PHP com o seguinte código:

```
$conn = ftp_connect("$host");
if (!$conn)
{
  echo 'Error: Could not connect to ftp server<br />';
  exit;
}
echo "Connected to $host.<br />";
```

A chamada de função aqui é para `ftp_connect( )`. Essa função toma um nome de host como parâmetro e retorna um handle para a conexão ou `false` se a conexão não pôde ser estabelecida. A função também pode aceitar o número da porta no host a que se conectar como um segundo parâmetro opcional. (Não utilizamos isso aqui.) Se você não especificar um número de porta, por padrão é assumida a porta 21, que é padrão no FTP.

Efetuando logon no servidor FTP

O próximo passo é efetuar logon como um usuário particular com uma senha particular. Você pode alcançar isso utilizando a função `ftp_login( )`:

```
@ $result = ftp_login($conn, $user, $pass);
if (!$result)
{
  echo "Error: Could not log on as $user<br />";
  ftp_quit($conn);
  exit;
}
echo "Logged in as $user<br />";
```

A função recebe três parâmetros: uma conexão de FTP (obtida de `ftp_connect( )`), um nome de usuário e uma senha. Essa função retornará `true` se o usuário puder efetuar logon e `false`, caso contrário. Você notará que colocamos um símbolo `@` no início da linha para suprimir erros. Fazemos isso porque, se o usuário não puder efetuar logon, ele obterá um aviso do PHP na janela de navegador. Você pode capturar o erro da maneira como capturamos aqui, testando `$result` e fornecendo sua própria mensagem de erro mais amigável ao usuário.

Note que se a tentativa de login falha, realmente fechamos a conexão de FTP utilizando `ftp_quit( )` – veremos mais sobre isso logo adiante.

Verificando as horas de atualização do arquivo

Considerando que estamos atualizando uma cópia local de um arquivo, é sensato verificar se o arquivo precisa pré-atualizar primeiro, porque você não quer ter de refazer o download de um arquivo, particularmente um arquivo grande, se ele estiver atualizado. Isso evitará tráfego desnecessário de rede. Vejamos o código que verifica os horários de atualização de arquivos.

Os horários dos arquivos são a razão pela qual usamos funções FTP em vez de uma chamada muito mais simples a uma função de arquivo. As funções de arquivo podem facilmente ler e, em alguns casos, gravar arquivos sobre interfaces de rede, mas a maioria das funções de status, como `filemtime( )`, não funcionam remotamente. Isso mudará no futuro. As funções de status do sistema de arquivos são planejadas para suportar `ftp//:` no PHP5.1. Nesse estágio, podemos reimplementar esse script usando um simples `copy($remoteurl, $localurl)`.

Para decidir se precisamos fazer o download de um arquivo, verificamos se temos uma cópia local do arquivo, utilizando a função `file_exists( )`. Se não tivermos, então obviamente precisamos fazer download do arquivo. Se ele realmente existir, obtemos o horário da última modificação do arquivo utilizando a função `filemtime( )` e a armazenamos na variável `$localtime`. Se ele não existir, configuramos a variável `$localtime` como `0` para que ela seja "mais antiga" que qualquer data/hora de modificação de arquivo remoto possível:

```
echo 'Checking file time...<br />';
if (file_exists($localfile))
{
  $localtime = filemtime($localfile);
  echo 'Local file last updated ';
  echo date('G:i j-M-Y', $localtime);
  echo '<br />';
}
else
  $localtime=0;
```

(Você pode ler mais sobre as funções file_exists( ) e filemtime( ) nos Capítulos 2 e 18, respectivamente.)

Depois que obtemos a data/hora local, precisamos obter a data/hora de modificação do arquivo remoto. Você pode obter isso utilizando a função ftp_mdtm( ):

```
$remotetime = ftp_mdtm($conn, $remotefile);
```

Essa função aceita dois parâmetros – o handle de conexão de FTP e o caminho para o arquivo remoto – e retorna o registro de data/hora do UNIX da data/hora em que o arquivo foi modificado pela última vez, ou -1 se houver algum tipo de erro. Nem todos os servidores FTP suportam esse recurso, então talvez não obtenhamos um resultado útil da função. Nesse caso, escolhemos artificialmente configurar a variável $remotetime para ser mais "nova" do que a variável $localtime adicionando 1 a ela. Isso assegurará que seja feita uma tentativa para descarregar o arquivo:

```
if (!($remotetime >= 0))
{
  // Isso não significa que o arquivo não está lá, o servidor pode não suportar mod time
  echo 'Can't access remote file time.<br />';
  $remotetime=$localtime+1;  // certifica-se de uma atualização
}
else
{
  echo 'Remote file last updated ';
  echo date('G:i j-M-Y', $remotetime);
  echo '<br />';
}
```

Quando temos os dois horários, podemos compará-los para ver se precisamos ou não fazer download do arquivo:

```
if (!($remotetime > $localtime))
{
  echo 'Local copy is up to date.<br />';
  exit;
}
```

Fazendo download do arquivo

Nesta etapa, tentaremos fazer download do arquivo do servidor:

```
echo 'Getting file from server...<br />';
$fp = fopen ($localfile, 'w');
if (!$success = ftp_fget($conn, $fp, $remotefile, FTP_BINARY))
{
  echo 'Error: Could not download file';
  fclose($fp);
  ftp_quit($conn);
  exit;
}
fclose($fp);
echo 'File downloaded successfully';
```

Abrimos um arquivo local utilizando fopen( ) como vimos anteriormente. Depois que fizemos isso, chamamos a função ftp_fget( ), que tenta fazer download do arquivo e armazenamos em um arquivo local. Essa função recebe quatro parâmetros. Os primeiros três são simples e diretos – a conexão de FTP, o handle de arquivo local e o caminho para o arquivo remoto. O quarto parâmetro é o modo de FTP.

Há dois modos para uma transferência de FTP, ASCII e binário. O modo ASCII é utilizado para transferir arquivos de texto (isto é, arquivos que consistem unicamente em caracteres ASCII), e o

modo binário é utilizado para transferir tudo mais. A biblioteca de FTP do PHP vem com duas constantes predefinidas, FTP_ASCII e FTP_BINARY, que representam esses dois modos. Você precisa decidir qual modo é adequado ao seu tipo de arquivo e passar a constante correspondente para ftp_fget( ) como o quarto parâmetro. Nesse caso, estamos transferindo um arquivo zip e então utilizamos o modo FTP_BINARY.

A função ftp_fget( ) retorna true se tudo correr bem ou false se um erro for encontrado. Armazenamos o resultado em $success e deixamos que o usuário saiba o que aconteceu.

Depois que o download foi tentado, fechamos o arquivo local utilizando a função fclose( ).

Como uma alternativa para ftp_fget( ), poderíamos ter utilizado ftp_get( ), que tem o seguinte protótipo:

```
int ftp_get (int conexão_de_ftp, string caminho_para_o_arquivo_remoto,
string caminho_para_o_arquivo_remoto, int modo)
```

Essa função funciona de modo muito semelhante à função ftp_fget( ), mas não exige que o arquivo local esteja aberto. Você passa para ela o nome de arquivo de sistema do arquivo local que gostaria de gravar em vez de um handle de arquivo.

Note que não há equivalente do comando do FTP mget, que pode ser utilizado para fazer download de diversos arquivos por vez. Em vez disso, você deve fazer várias chamadas a ftp_fget( ) ou ftp_get( ).

Fechando a conexão

Depois que concluirmos a conexão de FTP, você deve fechá-la utilizando a função ftp_quit( ):

```
ftp_quit($conn);
```

Você deve passar a essa função o handle da conexão de FTP.

Fazendo o upload de arquivos

Se quiser adotar outro caminho, isto é, copiar arquivos a partir de seu servidor para uma máquina remota, você pode utilizar duas funções que são basicamente opostas de ftp_fget( ) e ftp_get( ). Essas funções são chamadas ftp_fput( ) e ftp_put( ). Essas funções têm os seguintes protótipos:

```
int ftp_fput (int conexão_de_ftp, string caminho_para_o_arquivo_remoto, int pa, int modo)

int ftp_put (int conexão_de_ftp, string caminho_para_o_arquivo_remoto,
             string caminho_do_arquivolocal, int modo)
```

Os parâmetros são os mesmos para os equivalentes de _get.

Evitando tempos limite

Um problema com que você talvez se depare ao fazer arquivos FTP é exceder o tempo máximo de execução. Você saberá se isso aconteceu porque o PHP apresentará uma mensagem de erro. Isso provavelmente ocorrerá especialmente se seu servidor estiver executando em uma rede lenta, congestionada, ou se você estiver fazendo download de um arquivo grande, como um clipe de filme.

O valor padrão do tempo de execução máximo para todos os scripts do PHP é definido no arquivo php.ini. Por padrão, é configurado em 30 segundos. Isso é projetado para capturar scripts que estiverem fora de controle. Entretanto, quando estiver transferindo arquivos por FTP, se seu link com o restante do mundo for lento ou se o arquivo for grande, a transferência de arquivos poderia muito bem levar mais tempo que isso.

Felizmente, podemos modificar o tempo de execução máximo para um script particular utilizando a função set_time_limit( ). Chamar essa função redefine o número máximo de segundos

que o script tem permissão de executar, iniciando a partir da hora em que a função é chamada. Por exemplo, se você chamar

```
set_time_limit(90);
```

então o script será capaz de executar outros 90 segundos a partir da hora em que a função é chamada.

Utilizando outras funções de FTP

Há várias outras funções de FTP úteis no PHP.

A função `ftp_size( )` pode informar o tamanho de um arquivo em um servidor remoto. Ela tem o seguinte protótipo:

```
int ftp_size(int conexão_de_ftp, string caminho_para_o_arquivo_remoto)
```

Essa função retorna o tamanho do arquivo remoto em bytes ou -1 se houver um erro. Isso não é suportado por todos os servidores de FTP.

Uma boa utilização de `ftp_size( )` é calcular o tempo máximo de execução a configurar para uma transferência particular. Dados o tamanho do arquivo e a velocidade de sua conexão, você pode fazer uma suposição sobre quanto tempo a transferência deve levar, e deve utilizar a função `set_time_limit( )` dessa maneira.

Você pode obter e exibir uma lista de arquivos em um diretório em um servidor FTP remoto com o seguinte código:

```
$listing = ftp_nlist($conn, "$directory_path");
foreach ($listing as $filename)
  echo "$filename <br />";
```

Esse código utiliza a função `ftp_nlist( )` para obter uma lista de nomes de arquivos em um diretório particular.

Em termos de outras funções de FTP, quase tudo que você pode fazer a partir de uma linha de comando do FTP, pode fazer com as funções de FTP. As funções específicas correspondentes a cada comando do FTP podem ser encontradas no manual PHP on-line em http://php.net/manual/en/ref.ftp.php.

A exceção é `mget` (get múltiplo), mas você pode utilizar `ftp_nlist( )` para obter uma lista de arquivos e então buscá-los conforme necessário.

Leitura adicional

Abordamos muitos tópicos neste capítulo e como você poderia esperar, há uma grande quantidade de material sobre esses tópicos. Para obter informações sobre os protocolos individuais e como eles funcionam, consulte as RFCs em http://www.rfc-editor.org/.

Encontre também algumas informações de seu interesse sobre protocolos no World Wide Web Consortium:

http://www.w3.org/Protocols/

Você também pode consultar um livro sobre TCP/IP como *Rede de Computadores* de Andrew Tannenbaum, publicado pela Campus.

A seguir

Entraremos no Capítulo 20 e examinaremos as bibliotecas de funções de data e calendário do PHP. Você verá como converter de formatos inseridos pelo usuário para formatos de PHP, para formatos do MySQL e de volta para seu formato original.

20

Gerenciando a data e a hora

NESTE CAPÍTULO, DISCUTIREMOS COMO VERIFICAR e formatar datas e horas e converter entre formatos de data. Isso é especialmente importante ao converter entre formatos de data do MySQL e do PHP, formatos de data do Unix e do PHP e datas inseridas pelo usuário em um formulário de HTML.

Os tópicos-chave deste capítulo incluem:

- Obtendo a data e a hora no PHP;
- Convertendo entre formatos de data do PHP e do MySQL;
- Calculando datas;
- Utilizando funções de calendário.

Obtendo a data e a hora a partir do PHP

No Capítulo 1, conversamos sobre como utilizar a função `date( )` para obter e formatar a data e a hora do PHP. Conversaremos agora sobre essa e algumas outras funções de data e hora do PHP um pouco mais detalhadamente.

Utilizando a função date( )

Como talvez você lembre, a função `date( )` aceita dois parâmetros, um deles opcional. O primeiro é uma string de formato e o segundo, que é opcional, é um registro de data/hora do Unix. Se você não especificar um registro de data/hora, então `date( )` assumirá como padrão a data/hora atual. Essa função retorna uma string formatada para representar a data apropriada.

Uma chamada típica à função de data poderia ser:

```
echo date('jS F Y');
```

Isso produzirá uma data do formato `29th October 2000`.

Os códigos de formato aceitos por `date( )` estão listados na Tabela 20.1.

Tabela 20.1 **Códigos de formato para a função date( ) do PHP**

Código	Descrição
a	Antes ou depois do meio-dia, representados como dois caracteres em letras minúsculas, am ou pm.
A	Antes ou depois do meio-dia, representados como dois caracteres em letras maiúsculas, AM ou PM.

Tabela 20.1 Continuação

Código	Descrição
B	Swatch Internet Time, um esquema universal de data/hora. Informações adicionais estão disponíveis em http://www.swatch.com/.
c	Data ISO 8601. Uma data é representada como YYYY-MM-DD. Um T maiúsculo separa a data da hora. A hora é representada como HH:MM:SS. Finalmente, o fuso horário é representado como um deslocamento da Hora Média de Greenwich (GMT – Greenwich Mean Time) – por exemplo, 2004-03-26T21 : 04 :42+11:00. (Este formato foi acrescentado no PHP5.)
d	Dia do mês como um número de 2 dígitos com um zero inicial. O intervalo é de 01 a 31.
D	Dia da semana no formato de texto abreviado em 3 caracteres. Intervalo é de Mon a Sun.
F	Mês do ano em formato de texto completo. Intervalo é de January a December.
g	Hora do dia em formato 12 horas sem zeros iniciais. O intervalo é de 1 a 12.
G	Hora do dia em formato 24 horas sem zeros iniciais. O intervalo é de 0 a 23.
h	Hora do dia no formato 12 horas com zeros iniciais. O intervalo é de 01 a 12.
H	Hora do dia no formato 24 horas com zeros iniciais. O intervalo é de 00 a 23.
i	Minutos depois da hora com zeros iniciais. O intervalo é de 00 a 59.
I	Horários especiais de verão/inverno, representados como um valor booleano. Isso retornará 1 se a data estiver no horário especial de verão/inverno e 0 se não estiver.
j	Dia do mês como um número sem zeros iniciais. O intervalo é de 1 a 31.
l	Dia da semana no formato completo de texto. O intervalo é de Monday a Sunday.
L	Ano bissexto, representado como um valor booleano. Isso retornará 1 se a data estiver em um ano bissexto e 0 se não estiver.
m	Mês do ano com um número de 2 algarismos com zeros iniciais. O intervalo é de 01 a 12.
M	Mês do ano no formato de texto abreviado de 3 caracteres. O intervalo é de Jan a Dec.
n	Mês do ano como um número sem zeros iniciais. O intervalo é de 1 a 12.
O	A diferença em horas entre o fuso horário atual e a Hora Média de Greenwich (GMT – Greenwich Mean Time), por exemplo +1600.
r	A RFC822 formatou a data e a hora, por exemplo, Wed, 9 Oct 2002 18:45:30 +1600. (Acescentado no PHP 4.0.4.)
s	Segundos depois do minuto com zeros iniciais. O intervalo é de 00 a 59.
S	Sufixo ordinal para datas no formato de 2 caracteres. Isso pode ser st, nd, rd ou th dependendo do número que acompanha.
t	Número total de dias no mês da data. O intervalo é de 28 a 31.
T	Configuração de fuso horário do servidor no formato de 3 caracteres, por exemplo, EST.
U	Número total de segundos desde 1º de janeiro de 1970 até o momento atual; também conhecido como o registro de data/hora do Unix para essa data.
w	Dia da semana como um único dígito. O intervalo é de 0 (domingo) a 6 (sábado).
W	O número de semanas no ano, compatível com o ISO-8601. (Acrescentado no PHP 4.1.0.)
y	Ano no formato de 2 algarismos, por exemplo, 00.
Y	Ano no formato de 4 algarismos, por exemplo, 2000.
z	Dia do ano como um número. O intervalo é de 0 a 365.
Z	Deslocamento em relação ao fuso horário atual em segundos. O intervalo é -43200 a 43200.

Lidando com registros de data/hora do Unix

O segundo parâmetro para a função `date( )` é um registro de data/hora do Unix. Em caso de você estar querendo saber exatamente o que isso significa, a maioria dos sistemas de Unix armazena a data e a hora atuais como um inteiro de 32 bits contendo o número de segundos desde a meia-noite de 1º de janeiro de 1970, GMT, também conhecido como a Unix Epoch. Isso pode parecer um pouco esotérico se você não estiver familiarizado com o assunto, mas é um padrão e os computadores lidam facilmente com inteiros.

Os registros de data/hora (timestamps) do Unix são uma maneira compacta de armazenar uma data e hora, mas vale notar que eles não sofrem do problema do ano 2000 (Y2K) que afeta alguns outros formatos de data abreviados ou compactos. Eles têm problemas parecidos, porém, porque representam apenas uma gama limitada de horário usando um inteiro de 32 bits. Se seu software precisa lidar com eventos antes de 1902 ou depois de 2038, você terá problemas.

Você não precisa se preocupar sobre seu software até ser usado em 2038. Os registros de data/hora não têm um tamanho fixo, mas são associados ao tamanho de um long do C, que tem pelo menos 32 bits. Se seu software ainda estiver em uso em 2038, é muito provável que seu compilador estará usando um tipo maior.

Apesar de ser uma convenção padrão UNIX, esse é ainda o formato que é utilizado por `date( )` e várias outras funções do PHP mesmo se você estiver executando em um Windows. A única diferença é que, para Windows, o registro de data/hora tem de ser positivo.

Se quiser converter uma data/hora *para* um registro de data/hora do Unix, você pode utilizar a função `mktime( )`, que tem o seguinte protótipo:

```
int mktime ([int hora[, int minuto[, int segundo[, int mês[,
         int dia[, int ano [, int is_dst]]]]]]])
```

Os parâmetros são relativamente auto-explicativos, com a exceção do último, `é_dst`, que representa se a data está ou não no horário de verão. Você pode configurar isso como 1 se estiver, 0 se não estiver, ou -1 (o valor padrão) se você não souber. Isso é opcional; então, de qualquer jeito, você raramente irá utilizá-lo.

A principal armadilha a evitar com essa função é que os parâmetros estão em uma ordem relativamente não-intuitiva. A ordenação não serve para omitir a hora. Se não estiver preocupado com a hora, você pode passar 0s para os parâmetros *hora*, *minuto* e *segundo*. Mas você pode omitir valores do lado direito da lista de parâmetro. Se você deixar parâmetros em branco, eles serão configurados como os valores atuais. Portanto, uma chamada como

```
$timestamp = mktime( );
```

retornará o registro de data/hora do Unix para a data e a hora atuais. Você poderia também, naturalmente, obter isso chamando:

```
$timestamp = time( );
```

A função `time( )` não tem nenhum parâmetro e sempre retorna o registro de data/hora do Unix para a data e hora atuais.

Outra opção é a função `date( )`, como já foi discutido. A string de formato "U" exige um registro de data/hora. A instrução a seguir é equivalente as duas anteriores:

```
timestamp = date("U")
```

Você pode passar um ano de 2 ou 4 algarismos para `mktime( )`. Valores de dois dígitos de 0 a 69 serão interpretados como os anos 2000 a 2069 e valores de 70 a 99 serão interpretados como 1970 a 1999.

Aqui estão alguns exemplos para ilustrar o uso de `mktime( )`:

```
$time = mktime(12, 0, 0);
```

fornece o meio-dia da data de hoje.

```
$time = mktime(0,0,0,1,1);
```

fornece 1º de janeiro do ano atual.

Você também utilizar `mktime( )` para aritmética de data simples. Por exemplo,

```
$time = mktime(12,0,0,$mon,$day+30,$year);
```

adiciona 30 dias à data especificada nos componentes, embora (`$day+30`) seja geralmente mais do que o número de dias naquele mês.

Para eliminar alguns problemas com hora, use hora 12 em vez de hora 0. Se você adicionar (24 * 60 * 60) a meia-noite em um dia de 25 horas, será o mesmo dia. Adicione o mesmo número a meio-dia, e será `11am`, mas pelo menos será o dia correto.

Utilizando a função **getdate( )**

Outra função para determinar data que você talvez ache útil é a função `getdate( )`. Essa função tem o seguinte protótipo:

```
array getdate (int timestamp)
```

Ela aceita um registro de data/hora como parâmetro e retorna um array associativo representando as partes dessa data e hora como mostra a Tabela 20.2.

Tabela 20.2 **Pares de chave-valor de array associativo a partir da função getdate( )**

Chave	Valor
`seconds`	Segundos, numérico
`minutes`	Minutos, numérico
`hours`	Horas, numérico
`mday`	Dia do mês, numérico
`wday`	Dia da semana, numérico
`mon`	Mês, numérico
`year`	Ano, numérico
`yday`	Dia do ano, numérico
`weekday`	Dia da semana, formato completo de texto
`month`	Mês, formato de texto completo
`0`	Registro de data/hora, numérico

Depois de ter essas partes em um array, você pode facilmente processá-las em qualquer formato requerido. O elemento `0` no array (o registro de data/hora) pode parecer inútil, mas se você chamar `getdate( )` sem um parâmetro, ele dará o registro de data/hora atual.

Validando datas

Você pode utilizar a função `checkdate( )` para verificar se uma data é válida. Isso é especialmente útil para verificar datas de entrada de usuário. A função `checkdate( )` tem o seguinte protótipo:

```
int checkdate (int month, int day, int year)
```

Ela verificará se o ano é um inteiro válido entre 0 e 32.767, se o mês é um inteiro entre 1 e 12 e se o dia fornecido existe nesse mês particular. A função leva em consideração os horários de verão.

Por exemplo,

```
checkdate(9, 18, 1972);
```

retornará true enquanto

```
checkdate(9, 31, 2000)
```

não retornará.

Convertendo entre formatos de data do PHP e do MySQL

As datas e as horas no MySQL são recuperadas de maneira ligeiramente diferente da maneira como você poderia esperar. Os campos de data/hora funcionam de maneira relativamente normal, mas o MySQL espera que as datas sejam inseridas com o ano primeiro. Por exemplo, a data 29 de agosto de 2000 poderia ser inserida como 2000-08-29 ou como 00-08-29. As datas recuperadas a partir do MySQL também estarão nessa ordem por padrão.

Dependendo do público-alvo, você pode não achar essa função amigável ao usuário. Para se comunicar entre PHP e MySQL então, normalmente precisamos realizar alguma conversão de data. Isso pode ser feito em qualquer extremidade.

Ao colocar datas no MySQL a partir do PHP, você pode facilmente colocá-las no formato correto utilizando a função date( ) como mostrado anteriormente. Um pequeno cuidado é que você deve utilizar as versões do dia e mês com zeros iniciais para evitar confundir o MySQL. Você pode usar um ano com dois dígitos. Se escolher fazer a conversão no MySQL, duas funções úteis são DATE_FORMAT( ) e Unix_ TIMESTAMP( ).

A função DATE_FORMAT( ) funciona de maneira semelhante à função equivalente do PHP mas utiliza diferentes códigos de formato. Algo comum que queremos fazer é formatar uma data no formato MM-DD-AAAA em vez de no formato nativo AAAA-MM-DD do MySQL. Você pode fazer isso escrevendo sua consulta da seguinte maneira:

```
SELECT DATE_FORMAT(coluna_de_data, '%m %d %Y')
FROM nomedatabela;
```

O código de formato %m representa o mês como um número de 2 algarismos; %d, o dia como um número de 2 algarismos; e %Y, o ano como um número de 4 algarismos. Um resumo dos códigos de formato do MySQL mais úteis para esse propósito é mostrado na Tabela 20.3.

Tabela 20.3 **Códigos de formato para a função DATE_FORMAT( ) do MySQL**

Código	Descrição
%M	Mês, texto completo
%W	Nome do dia da semana, texto completo
%D	Dia de mês, numérico, com sufixo de texto (por exemplo, 1)
%Y	Ano, numérico, 4 algarismos
%y	Ano, numérico, 2 algarismos
%a	Nome do dia da semana, em 3 caracteres
%d	Dia de mês, zeros iniciais e numéricos
%e	Dia de mês, numérico, sem zeros iniciais
%m	Mês, numérico, zeros iniciais

Tabela 20.3 Continuação

Código	Descrição
%c	Mês, numérico, sem zeros iniciais
%b	Mês, texto, 3 caracteres
%j	Dia de ano, numérico
%H	Hora, relógio de 24 horas, zeros iniciais
%k	Hora, relógio de 24 horas, sem zeros iniciais
%h ou %I	Hora, relógio de 12 horas, zeros iniciais
%l	Hora, relógio de 12 horas, sem zeros iniciais
%i	Minutos, zeros iniciais, numéricos
%r	Hora, 12 horas (hh:mm:ss [AM|PM])
%T	Hora, 24 horas (hh:mm:ss)
%S ou %s	Segundos, numéricos, zeros iniciais
%p	AM ou PM
%w	Dia da semana, numérico, de 0 (domingo) a 6 (sábado)

A função `Unix_TIMESTAMP` funciona de maneira semelhante, mas converte uma coluna em um registro de data/hora do Unix. Por exemplo,

```
SELECT Unix_TIMESTAMP(coluna_de_data)
FROM nomedatabela;
```

retornará a data formatada como um registro de data/hora do Unix. Você pode então fazer como faria com ele no PHP.

Você pode facilmente realizar cálculos e comparações com o registro de data/hora do Unix. Lembre-se, no entanto, de que um registro de data/hora geralmente pode representar datas apenas entre 1902 e 2038, enquanto o tipo de data do MySQL possui um alcance maior.

Uma maneira prática de proceder é usar o registro de data/hora do Unix para cálculos de data e o formato de data padrão quando estiver apenas armazenando ou exibindo datas.

Calculando datas em PHP

A maneira simples de calcular o período de tempo entre duas datas no PHP é utilizando a diferença entre registros de data/hora do Unix. Utilizamos essa abordagem no script mostrado na Listagem 20.1.

Listagem 20.1 **`calc_age.php` – Script descobre a idade de uma pessoa com base na sua data de nascimento**

```
<?php
 // configura a data para cálculo
 $day = 18;
 $month = 9;
 $year = 1972;

  // lembre-se de que você precisa de bday como dia mês e ano
 $bdayunix = mktime ('', '', '', $month, $day, $year);
 // obtém ts unix para bday
```

Listagem 20.1 **Continuação**

```
$nowunix = time( ); // obtém ts unix para hoje
$ageunix = $nowunix - $bdayunix; // acha a diferença
$age = floor($ageunix / (365 * 24 * 60 * 60)); // converte de segundos para anos

echo "Age is $age";
?>
```

Nesse script, configuramos a data para calcular a idade. Em uma aplicação real, é possível que essas informações talvez venham de um formulário HTML. Começamos chamando `mktime( )` para calcular o registro de data/hora para o aniversário e o tempo atual:

```
$bdayunix = mktime ('', '', '', $month, $day, $year);
$nowunix = mktime( ); // obtém ts unix para hoje
```

Agora que essas datas estão no mesmo formato, simplesmente podemos subtraí-las:

```
$ageunix = $nowunix - $bdayunix;
```

Agora, a parte um pouco mais difícil – converter esse período de tempo de volta para uma unidade de medida mais amigável a humanos. Isso não é um registro de data/hora mas a idade da pessoa medida em segundos. Podemos converter de volta para anos dividindo pelo número de segundos em um ano. Então, arredondamos utilizando a função `floor( )` uma vez que não se diz que uma pessoa tem, por exemplo 20 anos, até o final do seu vigésimo ano:

```
$age = floor($ageunix / (365 * 24 * 60 * 60)); // converte de segundos para anos
```

Entretanto, observe que essa abordagem é um pouco falha porque é limitada pelo limite de registro de data/hora do Unix (geralmente inteiros de 32 bits). Datas de nascimento não são ideais para registros de data/hora. Esse exemplo funciona em todas as plataformas apenas para pessoas nascidas em 1970 e posteriormente. O Windows não pode gerenciar registros de data/hora anteriores a 1970. Mesmo assim, esse cálculo não é sempre preciso porque não permite anos bissextos e pode falhar se o nascimento da pessoa ocorrer na troca do horário de verão.

Calculando datas no MySQL

O PHP não possui muitas funções de manipulação de data embutidas. Obviamente, você pode escrever as suas próprias, mas garantindo que conte corretamente os anos bissextos e o horário de verão. Outra opção é fazer download de funções de terceiros. Você pode descobrir muitas notas de contribuição de usuários no manual do PHP, mas apenas algumas são bem elaboradas.

Uma opção que pode não parecer óbvia a princípio é usar o MySQL. Ele fornece um limite extensivo de funções de manipulação de datas que funcionam para horas fora do limite confiável do registro de data/hora do Unix. Você precisa se conectar a um servidor MySQL para executar uma consulta MySQL, mas não precisa usar dados do banco de dados.

A seguinte consulta adiciona um dia à data 28 de fevereiro de 1700, e retorna a data resultante:

```
select adddate('1700-02-28', interval 1 day)
```

O ano 1700 não é um ano bissexto, então o resultado é 1700-03-01 (ano, mês, dia).

Você pode descobrir uma sintaxe extensiva para descrever e modificar datas e horas descritas no manual do MySQL, em:

http://www.mysql.com/doc/en/Date_and_time_functions.html

Infelizmente, não há uma maneira simples de obter o número de anos entre duas datas; portanto, o exemplo da data de nascimento ainda é um pouco falho. É possível obter a idade de uma pessoa em dias muito facilmente, e a Listagem 20.2 converte a idade em anos de forma imprecisa.

Listagem 20.2 **mysql_calc_age.php – Utilizando o MySQL para calcular a idade de uma pessoa com base na data de nascimento**

```
<?php
// define data para cálculo
$day = 18;
$month = 9;
$year = 1972;

// formata a data de nascimento como uma data ISO 8601
$bdayISO = date("c", mktime (0, 0, 0, $month, $day, $year));

// usa a consulta mysql para calcular uma idade em dias
$db = mysqli_connect('localhost', 'user', 'pass');
$res = mysqli_query($db, "select datediff(now( ), '$bdayISO')");
$age = mysqli_fetch_array($res);

// converte a idade em dias para a idade em anos (aproximadamente)
echo "Age is ".floor($age[0]/365.25);
?>
```

Depois de formatar o dia do nascimento como um registro de data/hora ISO, passe a seguinte consulta ao MySQL:

```
select datediff(now( ), '1972-09-18T00:00:00+10:00')
```

A função MySQL `now( )` sempre retorna a data e a hora atuais. A função MySQL `datediff( )` (acrescentada na versão 4.1.1) subtrai uma data de outra e retorna a diferença em dias.

É importante notar que você não está selecionando dados de uma tabela ou mesmo escolhendo um banco de dados para ser usado nesse script, mas você não precisa fazer log in no servidor MySQL com um nome de usuário e senha válidos.

Como nenhuma função embutida específica está disponível para esses cálculos, uma consulta SQL para calcular o número exato de anos é razoavelmente complexa. Aqui, tomamos um atalho e dividimos a idade em dias por 365;25 para obtermos a idade em anos. Esse cálculo pode ficar com um ano a menos se feito no aniversário de alguém, dependendo de quantos anos bissextos tiver ocorrido na vida da pessoa.

Utilizando microssegundos

Para algumas aplicações, medir o tempo em segundos não é preciso o suficiente para ser considerado útil. Se você quiser medir períodos muito curtos, como o tempo que leva para executar uma parte ou todo um script do PHP, precisa utilizar a função `microtime( )`.

Se estiver utilizando o PHP5, chame `microtime( )` com o parâmetro `get_as_float` definido para `true`. Essa chamada retorna um registro de data/hora como número de ponto flutuante pronto para qualquer uso desejado. Esse registro de data/hora é o mesmo retornado por `mktime( )`, `time( )` ou `date( )`, mas possui um componente fracional.

A instrução

```
echo number_format(microtime(true), 10, '.', '');
```

produz algo como 1080303003.1321949959.

Em versões anteriores, você não pode pedir o resultado como flutuante. Ele é fornecido como string. Uma chamada a `microtime( )` sem um parâmetro retorna uma string deste formato `"0.02149300 1080302326"`. O primeiro número é a parte fracionada, e o segundo é o número de segundos desde 1º de janeiro de 1970.

É mais útil lidar com números em vez de strings; portanto, se você não se importar de seu código exigir o PHP 5.0 para ser executado, é mais fácil chamar `microtime( )` com o parâmetro `true`.

Utilizando funções de calendário

O PHP tem um conjunto de funções que permite converter entre diferentes sistemas de calendário. Os principais calendários com que você trabalhará são o gregoriano, o juliano e o de contagem de dias do calendário juliano (Julian Day Count).

O calendário gregoriano é o que a maioria dos países ocidentais utiliza atualmente. A data 15 de outubro de 1582 no calendário gregoriano é equivalente a 5 de outubro de 1582 no calendário juliano. Antes dessa data, o calendário juliano era comumente utilizado. Países diferentes converteram para o calendário gregoriano em momentos diferentes e alguns não o fizeram até o início do século XX.

Embora talvez você já tenha ouvido a respeito desses dois calendários, talvez não tenha ouvido falar do calendário de contagem de dias do calendário juliano. Sob vários aspectos, isso é semelhante a um registro de data/hora do Unix. É uma contagem do número de dias desde uma data em torno de 4000 AC. Por si só, não é particularmente útil, mas é útil para converter entre formatos. Para converter de um formato para outro, você primeiro converte para um calendário de contagem de dias juliano e então para o calendário de saída desejado.

Para utilizar essas funções sob o Unix, você precisará ter compilado a extensão de calendário no PHP. Eles são integrados na instalação padrão do Windows.

Para dar a você uma prévia dessas funções, considere os protótipos para as funções que você utilizaria para converter do calendário gregoriano para o juliano:

```
int gregoriantojd (int month, int day, int year)
string jdtojulian(int julianday)
```

Para converter uma data, precisaríamos chamar estas duas funções:

```
$jd = gregoriantojd (9, 18, 1582);
echo jdtojulian($jd);
```

Isso ecoa a data juliana em um formato mm/dd/aaaa.

Essas variações de funções existem para converter entre os calendários gregoriano, juliano, francês, judaico e os registros de data/hora do Unix.

Leitura adicional

Se quiser ler mais sobre funções de data e hora no PHP e MySQL, você pode consultar as seções relevantes dos manuais em:

http://php.net/manual/en/ref.datetime.php

http://www.mysql.com/doc/en/Date_and_time_functions.html

Se você estiver convertendo entre calendários, consulte a página do manual para funções de calendário do PHP:

http://php.net/manual/en/ref.calendar.php

A seguir

Uma das únicas coisas úteis que você pode fazer com o PHP é criar imagens instantaneamente. O Capítulo 21 discute como utilizar as funções de biblioteca de imagem para alcançar alguns efeitos interessantes e úteis.

21

Gerando imagens

UMA DAS FUNÇÕES MAIS ÚTEIS DO PHP é a criação de imagens instantaneamente. O PHP tem algumas funções predefinidas de informações sobre imagem e você também pode utilizar a biblioteca GD22 para criar novas imagens ou manipular imagens existentes. Este capítulo discute como utilizar as funções de imagem para alcançar alguns efeitos úteis interessantes.

Os tópicos-chave deste capítulo incluem:

- Configurando suporte a imagens em PHP;
- Entendendo formatos de imagem;
- Criando imagens;
- Utilizando imagens geradas automaticamente em outras páginas;
- Utilizando texto e fontes para criar imagens;
- Desenhando figuras e plotando dados.

Especificamente, veremos dois exemplos: gerar instantaneamente botões de Web site e desenhar um gráfico de barras utilizando figuras de um banco de dados do MySQL.

Utilizamos a biblioteca GD22 aqui, mas há uma outra biblioteca de imagens PHP popular. A biblioteca ImageMagick não faz parte do PHP padrão mas pode ser facilmente instalada a partir da PECL (PHP Extension Class Library). ImageMagick e GD22 possuem muitos recursos bastante semelhantes, mas em algumas áreas a ImageMagick vai além. Se você quiser criar GIFs (até GIFs animados), deve examinar a ImageMagick. Se quiser trabalhar com imagens true color ou renderizar efeitos transparentes, compare as duas bibliotecas.

Consulte PECL para obter o download de ImageMagick para PHP em http://pecl.php.net/package/imagick.

Consulte o site principal da ImageMagick para obter demonstrações das capacidades e documentação detalhada em http://www.imagemagick.org.

Configurando suporte a imagens no PHP

Algumas das funções de imagem no PHP estão sempre disponíveis, mas a maioria delas precisa da biblioteca GD22. Para obter informações detalhadas sobre GD22, pesquise em:

http://www.boutell.com/GD2/

Desde o PHP 4.3, o PHP vem com sua própria versão da biblioteca GD2, suportada pela equipe do PHP. Essa versão é mais fácil de instalar com PHP e geralmente é mais estável; portanto, é aconselhável utilizá-la. Sob Windows, PNGs e JPEGs são automaticamente suportados.

Se você tem Unix e deseja trabalhar com PNGs, precisa instalar libpng em http;//www.libpng.org/pub/png/ e zlib em http://www.gzip.org/zlib/.

Você então precisará configurar o PHP com as seguintes opções:

```
--with-png-dir=/path/to/libpng
--with-zlib-dir=/path/to/zlib
```

Se tiver Unix e também quiser trabalhar com o JPEGs, você precisará fazer o download do jpeg-6b e recompilar o GD com suporte de JPEG incluído. Você pode fazer download dessa biblioteca em:

ftp://ftp.uu.net/graphics/jpeg/

Você então precisará reconfigurar o PHP com a opção:

```
--with-jpeg-dir=/path/to/jpeg-6b
```

e recompilá-lo.

Se quiser utilizar fontes TrueType nas imagens, você também precisará da biblioteca FreeType. Essa biblioteca também vem com o PHP4. Alternativamente, você pode fazer download dessa biblioteca em:

http://www.freetype.org/

Se, em vez disso, quiser utilizar as fontes PostScript Type 1, você precisará fazer download do t1lib, disponível em:

ftp://sunsite.unc.edu/pub/Linux/libs/graphics/

Você então precisará executar o PHP para configurar o programa com:

```
--with-t1lib[=caminho/para/t1lib]
```

Finalmente, você precisará, é claro, configurar o PHP utilizando --with-gd.

Entendendo formatos de imagem

A biblioteca GD suporta os formatos JPEG, PNG e WBMP. Ela não suporta mais o formato GIF. Vejamos brevemente cada um desses formatos.

JPEG

O JPEG (pronuncia-se "jota-pegue") na realidade significa *Joint Photographic Experts Group* e é o nome de um corpo de padrões. O formato de arquivo a que nos referirmos como JPEGs chama-se na realidade JFIF, que corresponde a um dos padrões emitidos pelo JPEG.

No caso de você não estar familiarizado com eles, os JPEGs são normalmente utilizados para armazenar imagens fotográficas ou outras imagens com muitas cores ou graduações de cor. Esse formato utiliza compactação com perdas – isto é, a fim de compactar uma fotografia em um arquivo menor, um pouco da qualidade de imagem é perdida. Como os JPEGs devem conter o que são imagens essencialmente analógicas, com graduações de cor, o olho humano pode tolerar alguma perda de qualidade. Esse formato não é adequado para desenhos a traço, texto ou blocos de cor sólida.

Você pode ler mais sobre JPEG/JFIF no site JPEG oficial:

http://www.jpeg.org/

PNG

PNG (pronuncia-se "ping")significa *Portable Network Graphics*. Esse formato de arquivo é o substituto do *GIF (Graphics Interchange Format)* por razões que discutiremos logo a seguir. O Web site do PNG descreve-o como "um formato de imagem poderoso com compactação sem perdas". Como não tem perda, esse formato de imagem é adequado para imagens que contém texto, linhas retas e blocos simples de cor como títulos e botões de Web site – todos os mesmos propósitos pelos quais você talvez já tenha utilizado o GIF antes.

Uma versão compactada PNG da mesma imagem geralmente é parecida em tamanho a uma versão compactada GIF. PNG também oferece transparência variável, correção de gama e entrelaçamento bidimensional. Entretanto, não suporta animações – para isso você deve utilizar o formato de extensão MNG, que ainda está em desenvolvimento.

Esquemas de compactação sem perda são bons para ilustrações mas geralmente não são uma boa forma de armazenar fotos grandes porque tendem a ter tamanhos de arquivo grandes.

Você pode ler mais sobre PNG no site oficial:

http://www.libpng.org/pub/png/

WBMP

WBMP significa *Wireless Bitmap*. Ele é um formato de arquivo projetado especificamente para dispositivos sem fio. Não é muito usado.

GIF

GIF significa *Graphics Interchange Format*. Esse é um formato sem perda compactado amplamente e utilizado na Web para armazenar imagens contendo texto, linha reta e bloco de cor única.

A pergunta que você provavelmente fará é: por que GD não suporta GIFs?

A resposta é que ele utilizava, até a versão 1.3. Se quiser instalar e utilizar as funções GIF em vez das funções PNG, você pode fazer download da versão GD 1.3 em:

http://www.linuxguruz.org/downloads/gd1.3.tar.gz

Note, porém, que os criadores do GD desencorajam a utilização dessa versão e não a suportam mais. Essa cópia da versão GIF talvez não esteja disponível eternamente.

Há uma boa razão para o GD não mais suportar GIFs. Os GIFs padrão utilizam uma forma de compactação conhecida como *LZW (Lempel Ziv Welch)*, que está sujeita a uma patente que pertence à UNISYS. Depois de permitir que o formato se estabelecesse como um padrão por muitos anos, a UNISYS decidiu que fornecedores de progrmas quem lêem e gravam GIFs têm de pagar taxas de licença à UNISYS. Por exemplo, a Adobe pagou uma taxa de licença por produtos como o Photoshop que são utilizados para criar GIFs. As bibliotecas de código parecem estar na situação em que os escritores da biblioteca de código devem pagar uma taxa, e, além disso, os usuários da biblioteca também devem pagar uma taxa. Portanto, se utilizar uma versão GIF da biblioteca GD no seu Web site, você talvez deva à UNISYS algumas taxas de licença relativamente pesadas. A funcionalidade de leitura de GIF está fora da licença e, assim, ainda está disponível.

Essa situação é lamentável porque os GIFs estavam em utilização durante muitos anos antes de a UNISYS escolher impor a licença. Portanto, o formato tornou-se um dos padrões para a Web. Uma forte animosidade existe contra a patente na comunidade de desenvolvimento Web. Você pode ler sobre isso (e formar sua própria opinião) no site da UNISYS :

http://www.unisys.com/about__unisys/lzw/

e em Burn All Gifs, a oposição, em:

http://burnallgifs.org/

A versão norte-americana da patente LZW expirou em 21 de junho de 2003, e a européia e asiática em momentos diferentes de 2004. Não somos advogados, e nada disso deve ser interpretado como aconselhamento jurídico, mas consideramos mais fácil usar PNGs, independente da diretiva. O suporte do navegador para PNGs era variável alguns anos atrás, mas é bom em todos os navegadores recentes. O suporte avançado ao recurso PNG ainda é falho, mas imagens simples devem funcionar em todo lugar.

Espera-se que (neste momento) o suporte a GIF seja restaurado no PHP 5.0.1 e PHP 4.3.8 (que estão previstos para serem lançado depois de expirar a patente de LZW).

Criando imagens

Os quatro passos básicos para criar uma imagem no PHP são:

1. Criar uma imagem de tela em que trabalhar.
2. Desenhar formas ou imprimir texto nessa tela.
3. Gerar a saída do gráfico final.
4. Limpar os recursos.

Iniciaremos com um script muito simples de criação de imagem. Esse script é mostrado na Listagem 21.1.

Listagem 21.1 **simplegraph.php – Envia para a saída um gráfico linear simples com o rótulo Sales**

```
<?php
// configura a imagem
  $height = 200;
  $width = 200;
  $im = ImageCreate($width, $height);
  $white = ImageColorAllocate ($im, 255, 255, 255);
  $black = ImageColorAllocate ($im, 0, 0, 64);

// desenha na imagem
  ImageFill($im, 0, 0, $blue);
  ImageLine($im, 0, 0, $width, $height, $white);
  ImageString($im, 4, 50, 150, 'Sales', $white);

// envia a imagem para a saída
  Header ('Content-type: image/png');
  ImagePng ($im);

// limpa
  ImageDestroy($im);
?>
```

A saída de executar esse script é mostrada na Figura 21.1.

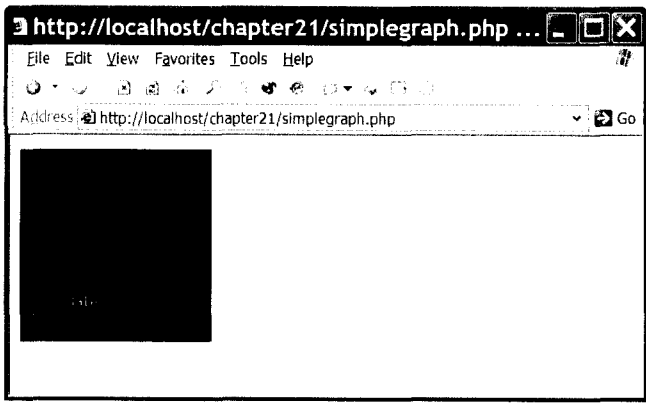



Figura 21.1 O script desenha um fundo preto e então adiciona uma linha e um rótulo de texto à imagem.

Guiaremos você pelos passos necessários para criar essa imagem.

Criando uma imagem de tela

Para começar a construir ou alterar uma imagem no PHP, você precisará criar um identificador de imagem. Há duas maneiras básicas de fazer isso. Uma é criar uma tela em branco, com a qual você pode fazer uma chamada para a função `ImageCreate( )`, como fizemos nesse script com o seguinte:

```
$im = ImageCreate($width, $height);
```

Você precisa passar dois parâmetros para `ImageCreate( )`. O primeiro é a largura da nova imagem e o segundo é a altura. A função retornará um identificador para a nova imagem. Esses funcionam de modo bastante parecido com os handles de arquivos.

Uma maneira alternativa é ler em um arquivo de imagem existente que você então possa filtrar, redimensionar ou adicionar. Você pode fazer isso com uma das funções `ImageCreateFromPNG( )`, `ImageCreateFromJPEG( )` ou `ImageCreateFromGIF( )` dependendo do formato de arquivo em que estiver lendo. Cada uma dessas toma o nome de arquivo como um parâmetro, como em, por exemplo:

```
$im = ImageCreateFromPNG('baseimage.png');
```

Um exemplo é mostrado mais adiante neste capítulo utilizando imagens existentes para criar instantaneamente botões.

Desenhando ou imprimindo texto na imagem

Há realmente duas etapas para desenhar ou imprimir texto na imagem. Primeiro, você deve selecionar as cores com que deseja desenhar. Como você provavelmente já sabe, as cores que são exibidas em um monitor de computador são compostas de diferentes quantidades de luz azul, verde e vermelha. Os formatos de imagem utilizam uma paleta de cores que consiste em um subconjunto especificado de todas as possíveis combinações das três cores. Para utilizar uma cor para desenhar em uma imagem, adicione essa cor à paleta da imagem. Faça isso para cada cor que quiser utilizar, mesmo preto-e-branco.

Você pode selecionar cores para sua imagem chamando a função `ImageColorAllocate( )`. É necessário passar para seu identificador de imagens os valores vermelho, verde e azul (red, green e blue – RGB) da cor que você deseja utilizar na função.

Na Listagem 21.1, estamos utilizando duas cores: azul e branco. Alocamos essas cores chamando:

```
$white = ImageColorAllocate ($im, 255, 255, 255);
$blue = ImageColorAllocate ($im, 0, 0, 64);
```

A função retorna um identificador de cor que podemos utilizar para acessar a cor mais tarde.

Segundo, para realmente desenhar na imagem, várias funções diferentes estão disponíveis, dependendo do que você quer desenhar – linhas, arcos, polígonos ou texto.

As funções de desenho geralmente exigem os seguintes itens como parâmetros:

- O identificador de imagem;
- As coordenadas iniciais e às vezes as coordenadas finais do que você quer desenhar;
- A cor com que você quer desenhar;
- Para texto, as informações de fonte.

Nesse caso, utilizamos três funções de desenho. Vamos examinar uma por vez.

Primeiro, pintamos um fundo azul em que desenharemos utilizando a função `ImageFill( )`:

```
ImageFill($im, 0, 0, $blue);
```

Essa função toma o identificador de imagem, as coordenadas iniciais da área a pintar (X e Y) e a cor para preencher como parâmetros.

Nota É importante notar que as coordenadas da imagem começam no canto superior esquerdo, que é x=0, y=0. O canto inferior direito da imagem é x=$width, y=$height. Isso é o oposto das típicas convenções de representação gráfica, então tenha cuidado!

Em seguida, desenhamos uma linha do canto superior esquerdo (0, 0) para o canto inferior direito ($width, $height) da imagem:

```
ImageLine($im, 0, 0, $width, $height, $white);
```

Essa função toma o identificador de imagem, o ponto inicial x e y para a linha, o ponto terminal e então a cor, como parâmetros.

Por fim, adicionamos um rótulo ao gráfico:

```
ImageString($im, 4, 50, 150, 'Sales', $white);
```

A função `ImageString( )` aceita alguns parâmetros ligeiramente diferentes. O protótipo para essa função é:

```
int imagestring (resource im, int font, int  x, int  y, string s, int col)
```

Ela aceita como parâmetros o identificador de imagem, a fonte, as coordenadas x e y para começar a escrever o texto, o texto a ser escrito e a cor.

A fonte é um número entre 1 e 5. Essas representam um conjunto de fontes embutidas. Como uma alternativa a isso, você pode utilizar fontes TrueType ou PostScript Type 1. Cada uma dessas fontes tem um conjunto de funções correspondentes. Utilizaremos as funções TrueType no próximo exemplo.

Uma boa razão para utilizar um dos conjuntos de função alternativos é que o texto escrito pela função `ImageString( )` e funções associadas, como `ImageChar( )` (escrever um caractere para a imagem), não é suavizado (*aliased*). As funções TrueType e PostScript produzem texto suavizado (*anti-aliased*).

Se você não estiver seguro sobre qual é a diferença, veja a Figura 21.2. Onde aparecem curvas ou linhas com ângulos nas letras, o texto não suavizado aparece serrilhado. A curva ou ângulo é alcançado utilizando um efeito de "escada". Na imagem suavizada, quando houver curvas ou ângulos no texto, os pixels nas cores entre o fundo e a cor de texto serão utilizados para suavizar a aparência do texto.

Normal

Anti-aliased

Figura 21.2 Texto normal aparece serrilhado, especialmente em um tamanho grande da fonte. A suavização (anti-aliasing) atenua curvas e cantos de letras.

Enviando para a saída a imagem gráfica final

Você pode enviar a saída de uma imagem diretamente para o navegador ou para um arquivo.

Neste exemplo, enviamos a saída da imagem para o navegador. Esse é um processo de duas etapas. Primeiro, precisamos dizer ao navegador Web que enviamos para a saída uma imagem em vez do texto ou HTML. Fazemos isso utilizando a função `Header( )` para especificar o tipo MIME da imagem:

```
Header ('Content-type: image/png');
```

Normalmente quando você recupera um arquivo no navegador, o tipo MIME é a primeiro que o servidor Web envia. Para uma página HTML ou de PHP (execução post), envia-se primeiro:

```
Content-type: text/html
```

Isso instrui o navegador a interpretar os dados que seguem.

Nesse caso, queremos dizer ao navegador que estamos enviando uma imagem em vez da saída HTML normal. Podemos fazer isso utilizando a função `Header( )`, que ainda não discutimos.

Essa função envia strings brutas de cabeçalho de HTTP. Outra aplicação típica disso é fazer redirecionamentos de HTTP. Esses instruem o navegador a carregar uma página diferente em vez da página solicitada. Em geral, são utilizados quando uma página é movida. Por exemplo:

```
Header ('Location: http://www.domain.com/new_home_page.html');
```

Um ponto importante a notar ao utilizar a função `Header( )` é que ela não pode ser executada se um cabeçalho de HTTP já foi enviado para a página. O PHP enviará um cabeçalho de HTTP automaticamente para você logo que você enviar a saída de algo para o navegador. Portanto, se tiver alguma instrução `echo` ou mesmo algum espaço em branco antes de abrir sua tag de PHP, os cabeçalhos serão enviados e você obterá uma mensagem de aviso do PHP quando tentar chamar `Header( )`. Mas, você pode enviar diversos cabeçalhos de HTTP com diversas chamadas para a função `Header( )` no mesmo script, embora eles devessem aparecer antes de qualquer saída ser enviada para o navegador.

Depois que enviamos os dados de cabeçalho, geramos a saída dos dados da imagem com uma chamada a:

```
ImagePng ($im);
```

Isso envia a saída para o navegador no formato PNG. Se você quisesse enviar para a saída um formato diferente, poderia chamar `ImageJPEG( )` – se o suporte de JPEG estiver ativado. Você também precisaria enviar o cabeçalho correspondente primeiro; isto é:

```
Header ('Content-type: image/jpeg');
```

A segunda opção que você pode utilizar, como uma alternativa a todas as anteriores, é gravar a imagem em um arquivo em vez de no navegador. Você pode fazer isso adicionando o segundo parâmetro opcional a `ImagePNG( )` (ou uma função semelhante aos outros formatos suportados):

```
ImagePNG($im, $filename);
```

Lembre-se de que todas as regras normais sobre como gravar em um arquivo de PHP se aplicam (por exemplo, ter permissões configuradas corretamente).

Limpando

Quando terminar de fazer uma imagem, você deve retornar os recursos que utilizou para o servidor destruindo o identificador de imagem. Você pode fazer isso com uma chamada para `ImageDestroy( )`:

```
ImageDestroy($im);
```

Utilizando imagens automaticamente geradas em outras páginas

Como um cabeçalho pode ser enviado somente uma vez e essa é a única maneira de dizer ao navegador que estamos enviando dados de imagem, é ligeiramente difícil embutir quaisquer imagens que criamos instantaneamente em uma página regular. Três maneiras de fazer isso são as seguintes:

1. Fazer uma página inteira consistir em saída de imagem, como fizemos no exemplo anterior.
2. Gravar a imagem em um arquivo como anteriormente mencionado e então referenciá-la com uma tag <IMG> normal.
3. Colocar o script de produção de imagem em uma tag de imagem.

Já abordamos os métodos 1 e 2. Vamos ver brevemente o método 3 agora. Para utilizar esse método, você inclui a imagem in-line em HTML tendo uma tag de imagem junto com as seguintes linhas:

```
<img src="simplegraph.php" height="200" width="200" alt="Sales going down" />
```

Em vez de colocar diretamente em um PNG, JPEG ou GIF, coloque no script de PHP que gera a imagem na tag SRC. Isso será recuperado e a saída, adicionada in-line, como mostrado na Figura 21.3.

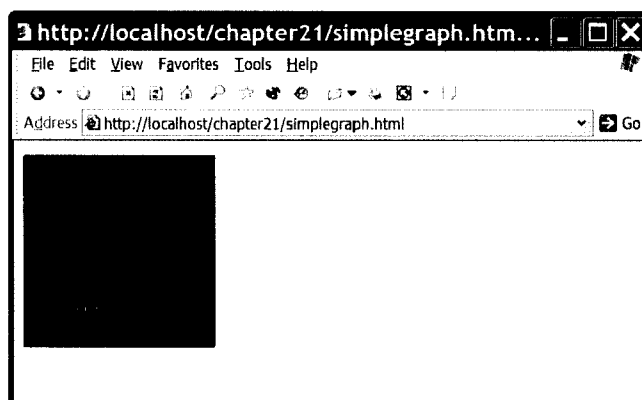



Figura 21.3 A imagem in-line produzida dinamicamente parece idêntica a uma imagem normal para o usuário final.

Utilizando texto e fontes para criar imagens

Veremos mais um exemplo complicado. É útil ser capaz de criar automaticamente botões ou outras imagens para Web site. Você pode construir botões simples baseados em um retângulo de cor de fundo utilizando as técnicas já discutidas. Também podemos gerar efeitos mais complicados programando, mas geralmente podemos fazer isso de uma forma mais fácil em um programa de desenho livre. Isso também facilita arranjar um artista para fazer o trabalho e deixar os programadores programarem.

Neste exemplo, geraremos botões utilizando um modelo de botão em branco que permite ter recursos como bordas em relevo e assim por diante, que são muito mais fáceis de gerar utilizando Photoshop, GIMP ou alguma outra ferramenta de imagem gráfica. Com a biblioteca de imagem no PHP, podemos iniciar com uma imagem de base e desenhar por cima dela.

Também utilizaremos as fontes TrueType para que possamos utilizar texto suavizado. As funções da fonte TrueType têm suas próprias peculiaridades, que discutiremos.

O processo básico é tomar algum texto e gerar um botão com esse texto. O texto será centralizado tanto horizontal como verticalmente no botão e será gerado no maior tamanho de fonte que couber no botão.

Construímos um front-end para o gerador de botão para teste e experimento. Essa interface é mostrada na Figura 21.4. (Não incluímos a HTML para esse formulário aqui, uma vez que ele é muito simples, mas você pode encontrá-lo no CD em `design_button.html`.)

Você poderia utilizar esse tipo de interface para um programa a fim de automaticamente gerar Web sites. Também poderia chamar o script que escrevemos de modo in-line, para gerar todos os botões do Web site instantaneamente.

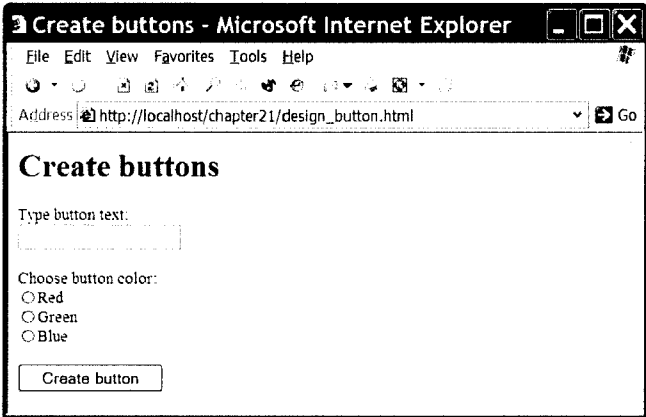



Figura 21.4 O front-end permite que um usuário escolha a cor de botão e digite o texto requerido.

A típica saída do script é mostrada na Figura 21.5.

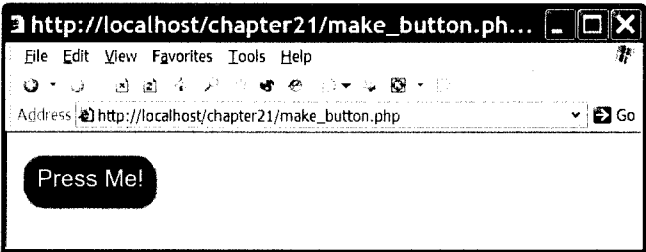



Figura 21.5 Um botão gerado pelo script `make_button.php`.

O botão é gerado por um script chamado `make_button.php`. Esse script é mostrado na Listagem 21.2.

Listagem 21.2 **`make_button.php` – Esse script pode ser chamado a partir do formulário no `design_button.html` ou de dentro de uma tag de imagens de HTML**

```
<?php
// verifica se temos os dados apropriados de variáveis
// as variáveis são o texto e a cor do botão

$button_text = $HTTP_POST_VARS['button_text'];
$color = $HTTP_POST_VARS['color'];

if (empty($button_text) || empty($color))
{
  echo 'Could not create image - form not filled out correctly';
  exit;
}

// cria uma imagem do fundo direito e verifica o tamanho
$im = imagecreatefrompng ($color.'-button.png');

$width_image = ImageSX($im);
$height_image = ImageSY($im);

// Nossas imagens precisam de uma margem de 18 de pixels a partir da borda da imagem
$width_image_wo_margins = $width_image - (2 * 18);
$height_image_wo_margins = $height_image - (2 * 18);
```

Listagem 21.2 **Continuação**

```
// Planeja se o tamanho da fonte se ajustará e o diminui até se ajustar
// Inicia com o maior tamanho que se ajustará razoavelmente aos nossos botões
$font_size = 33;

// você precisa informar à GD2 onde suas fontes residem
putenv('GDFONTPATH=C:\WINNT\Fonts');
$fontname = 'arial';

do
{
 $font_size--;

  // descobre o tamanho do texto nesse tamanho de fonte
  $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);

  $right_text = $bbox[2];   // coordenada direita
  $left_text = $bbox[0];    // coordenada esquerda
  $width_text = $right_text - $left_text;  // Qual é sua largura?
  $height_text = abs($bbox[7] - $bbox[1]);  // Qual é sua altura?

}
while ( $font_size>8 &&
       ( $height_text>$height_image_wo_margins ||
         $width_text>$width_image_wo_margins )
      );

if ( $height_text>$height_image_wo_margins ||
     $width_text>$width_image_wo_margins )
{
  // nenhum tamanho de fonte legível se ajustará no botão
  echo 'Text given will not fit on button.<br />';
}
else
{
  // Localizamos um tamanho da fonte que se ajuste
  // Agora planeja onde colocar

  $text_x = $width_image/2.0 - $width_text/2.0;
  $text_y = $height_image/2.0 - $height_text/2.0 ;

  if ($left_text < 0)
      $text_x += abs($left_text);    // adiciona fator ao deslocamento esquerdo

  $above_line_text = abs($bbox[7]);   // Quanto acima da linha de base?
  $text_y += $above_line_text;        // adiciona fator de linha de base

  $text_y -= 2;  // fator de ajustamento para forma do nosso exemplo

  $white = ImageColorAllocate ($im, 255, 255, 255);

  ImageTTFText ($im, $font_size, 0, $text_x, $text_y, $white, $fontname,
               $button_text);

  Header ('Content-type: image/png');
  ImagePng ($im);
}

ImageDestroy ($im);
?>
```

Esse é um dos scripts mais longos que vimos até agora. Vamos examiná-lo seção por seção. Iniciamos com alguma verificação de erros básica e então configuramos a tela em que trabalharemos.

Configurando a tela de base

Na Listagem 21.2, em vez de começar a partir do zero, iniciamos com uma imagem existente para o botão. Temos uma escolha de três cores no botão básico: vermelho (`red-button.png`), verde (green-button.png) e azul (`blue-button.png`).

A cor escolhida do usuário é armazenada na variável `$color` do formulário.

Iniciamos configurando um novo identificador de imagem baseado no botão apropriado:

```
$color = $_REQUEST['color'];
...
$im = ImageCreateFromPNG ($color.'-button.png');
```

A função `ImageCreateFromPNG( )` recebe o nome de um PNG como parâmetro e retorna um novo identificador de imagem para uma imagem contendo uma cópia desse PNG. Note que isso não modifica o PNG de base dessa maneira. Podemos usar as funções `ImageCreateFromJPEG( )` e `DImageCreateFromGIF( )` da mesma maneira se o suporte apropriado for instalado.

Nota A chamada a `ImageCreateFromPNG( )` só cria a imagem na memória. Para salvar a imagem em um arquivo ou dar saída para o navegador, devemos chamar a função `ImagePNG( )`. Veremos isso em breve, mas primeiro temos outro trabalho a fazer com nossa imagem.

Ajustando o texto no botão

Temos algum texto digitado pelo usuário armazenado na variável `$button_text`. O que queremos fazer é imprimir esse texto no botão no maior tamanho de fonte que se ajuste. Fazemos isso pela iteração, ou falando estritamente, pelo processo iterativo de tentativa e erro.

Iniciamos configurando algumas variáveis relevantes. As duas primeiras são a altura e a largura da imagem de botão:

```
$width_image = ImageSX($im);
$height_image = ImageSY($im);
```

As duas seguintes representam uma margem da borda do botão. Nossas imagens de botão são em relevo, então precisaremos deixar espaço para isso além das bordas do texto. Se você estiver utilizando imagens diferentes, esse número será diferente! Em nosso caso, a margem em cada lado é de aproximadamente 18 pixels.

```
$width_image_wo_margins = $width_image - (2 * 18);
$height_image_wo_margins = $height_image - (2 * 18);
```

Além disso, precisamos configurar o tamanho da fonte inicial. Iniciamos com 32 (na verdade 33, mas decrementaremos isso em um minuto) porque essa é aproximadamente a maior fonte que se ajustará no botão afinal de contas:

```
$font_size = 33;
```

Com GD2, você precisa dizer onde suas fontes residem configurando a variável de ambiente `GDFONTPATH` como segue:

```
putenv('GDFONTPATH=C:\WINNT\Fonts');
```

Também configuramos o nome da fonte que queremos utilizar. Vamos utilizar essa fonte com as funções TrueType, que procurará o arquivo de fonte na localização anterior e acrescentará `.ttf` (TrueType Font) ao nome do arquivo.

```
$fontname = 'arial';
```

Note que, dependendo do sistema operacional, você pode ter de adicionar `.ttf` ao fim do nome de fonte.

Se não tiver Arial (a fonte que utilizamos aqui) em seu sistema, você facilmente poderá alterar para outra fonte TrueType.

Agora fazemos um loop, decrementando o tamanho da fonte em cada iteração, até que o texto enviado se ajuste razoavelmente no botão:

```
do
{
  $font_size--;

  // descobre o tamanho do texto nesse tamanho de fonte
  $bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);

  $right_text = $bbox[2];   // coordenada direita
  $left_text = $bbox[0];    // coordenada esquerda
  $width_text = $right_text - $left_text;  // Qual é sua largura?
  $height_text = abs($bbox[7] - $bbox[1]);  // Qual ê sua altura?

}
while ( $font_size>8 &&
       ( $height_text>$height_image_wo_margins ||
         $width_text>$width_image_wo_margins )
     );
```

Esse código testa o tamanho do texto examinando o que é chamado o *quadro delimitador* do texto. Fazemos isso utilizando a função `ImageGetTTFBBox( )` que é uma das funções da fonte TrueType. Após termos calculado o tamanho, vamos imprimir o botão usando uma fonte TrueType (neste caso Arial, mas pode utilizar qualquer uma de que você goste) e a função `ImageTTFText( )`.

O quadro delimitador de uma parte de texto é a menor caixa que você poderia desenhar em torno do texto. Um exemplo de um quadro delimitador é mostrado na Figura 21.6.

Figura 21.6 As coordenadas do quadro delimitador são dadas em relação à linha de base. A origem das coordenadas é mostrada aqui como (0,0).

Para obter as dimensões da caixa, chamamos:

```
$bbox=imagettfbbox ($font_size, 0, $fontname, $button_text);
```

Essa chamada diz: "Para o tamanho da fonte `$font_size` dado, com texto inclinado em um ângulo de zero grau, utilizando a fonte TrueType Arial, informe as dimensões do texto em `$button_text`".

Observe que você realmente precisa passar o caminho para o arquivo contendo a fonte na função. Nesse caso, ele está no mesmo diretório que o script (o padrão), então não especificamos um caminho mais longo.

A função retorna um array contendo as coordenadas dos cantos do quadro delimitador. O conteúdo do array é mostrado na Tabela 21.1.

Tabela 21.1 **Conteúdo do array do quadro delimitador**

Índice de array	Conteúdo
0	Coordenada X, canto inferior esquerdo
1	Coordenada Y, canto inferior esquerdo
2	Coordenada X, canto inferior direito
3	Coordenada Y, canto inferior direito
4	Coordenada X, canto superior direito
5	Coordenada Y, canto superior direito
6	Coordenada X, canto superior esquerdo
7	Coordenada Y, canto superior esquerdo

Para lembrar qual é o conteúdo do array, simplesmente lembre-se de que a numeração começa no canto inferior esquerdo do quadro delimitador e segue seu caminho no sentido anti-horário.

Há algo difícil sobre os valores retornados a partir da função `ImageTTFBBox( )`. Eles são valores de coordenadas, especificadas a partir de uma origem. Entretanto, de maneira diferente das coordenadas para imagens, que são especificadas em relação ao canto superior esquerdo, esses valores são especificados em relação a uma linha de base.

Veja a Figura 21.6 novamente. Você verá que desenhamos uma linha ao longo da parte inferior da maior parte do texto. Isso é conhecido como *linha de base*. Algumas letras ficam abaixo da linha de base, como y nesse exemplo. Essas letras são chamadas *descendentes*.

O lado esquerdo da linha de base é especificado como a origem de medidas – isto é, coordenada X, 0 e coordenada Y, 0. As coordenadas acima da linha de base têm uma coordenada X positiva e as coordenadas abaixo da linha de base têm uma coordenada X negativa.

Além disso, talvez o texto realmente tenha valores coordenados que residam fora do quadro delimitador. Por exemplo, talvez o texto realmente inicie em uma coordenada X de –1.

Isso tudo quer dizer que é necessário cuidado ao realizar cálculos com esses números.

Elaboramos a largura e a altura do texto da seguinte maneira:

```
$right_text = $bbox[2];   // coordenada direita
$left_text = $bbox[0];    // coordenada esquerda
$width_text = $right_text - $left_text;  // Qual é sua largura?
$height_text = abs($bbox[7] - $bbox[1]);  // Qual é sua altura?
```

Depois que obtemos isso, testamos a condição do loop:

```
} while ( $font_size>8 &&
          ( $height_text>$height_image_wo_margins ||
            $width_text>$width_image_wo_margins )
        );
```

Estamos testando dois conjuntos de condições aqui. O primeiro é que a fonte é ainda legível – não há sentido em torná-la muito menor que 8 pontos porque o botão torna-se muito difícil de ler. O segundo conjunto de condições testa se o texto se ajustará dentro do espaço de desenho que temos para ele.

Em seguida, verificamos para ver se nossos cálculos iterativos localizaram ou não um tamanho aceitável de fonte e notificaremos um erro se não:

```
if ( $height_text>$height_image_wo_margins ||
     $width_text>$width_image_wo_margins )
{
  // nenhum tamanho de fonte legível se ajustará no botão
  echo 'Text given will not fit on button.<br />';
}
```

Posicionando o texto

Se tudo estiver certo, a seguir elaboramos uma posição de base para o início do texto. Esse é o ponto intermediário do espaço disponível.

```
$text_x = $width_image/2.0 - $width_text/2.0;
$text_y = $height_image/2.0 - $height_text/2.0 ;
```

Por causa das complicações com o sistema de coordenadas relativas da linha de base, precisamos adicionar alguns fatores de correção:

```
if ($left_text < 0)
    $text_x += abs($left_text);     // adiciona fator ao deslocamento esquerdo

$above_line_text = abs($bbox[7]);   // Quanto acima da linha de base?
$text_y += $above_line_text;        // adiciona fator de linha de base

$text_y -= 2;  // fator de ajustamento para forma do nosso exemplo
```

Esses fatores de correção permitem a linha de base e um pequeno ajuste porque nossa imagem é um pouco deslocada para cima.

Escrevendo o texto no botão

Depois disso, tudo é tranqüilo. Para configurar a cor de texto, que será branca:

```
$white = ImageColorAllocate ($im, 255, 255, 255);
```

Podemos então utilizar a função `ImageTTFText( )` para realmente desenhar o texto no botão:

```
ImageTTFText ($im, $font_size, 0, $text_x, $text_y, $white, $fontname,
              $button_text);
```

Essa função aceita muitos parâmetros. Na ordem, eles são o identificador de imagem, o tamanho da fonte em pontos, o ângulo em que desejamos desenhar o texto, as coordenadas Y e X iniciais do texto, a cor de texto, o arquivo de fonte, e, por fim, o texto real que entrará no botão.

Nota O arquivo de fonte precisa estar disponível no servidor e não é necessário na máquina do cliente porque ele o verá como uma imagem.

Concluindo

Por fim, podemos dar saída do botão para o navegador:

```
Header ('Content-type: image/png');
ImagePng ($im);
```

Então, é hora de limpar os recursos e terminar o script:

```
ImageDestroy ($im);
```

É isso aí! Se tudo correu bem, agora devemos ter um botão na janela de navegador parecido com o botão que você viu na Figura 21.5.

Desenhando e plotando dados

Na última aplicação, vimos imagens existentes e texto. Ainda não vimos um exemplo com desenho, então faremos isso agora.

Neste exemplo, executaremos uma pesquisa eleitoral no nosso Web site para testar que, usuários votarão em uma eleição fictícia. Armazenaremos os resultados da pesquisa eleitoral em um banco de dados do MySQL e desenharemos um gráfico de barras dos resultados utilizando as funções de imagem.

Essas funções são principalmente utilizadas para plotagem. Você pode plotar em gráficos os dados que quiser – vendas, hits Web ou o que vier à sua mente.

Para esse exemplo, gastamos alguns minutos configurando um banco de dados do MySQL chamado `poll`. Ele contém uma tabela chamada `poll_results`, que armazena os nomes dos candidatos na coluna `candidate` e o número de votos que eles receberam na coluna `num_votes`. Também criamos um usuário para esse banco de dados chamado `poll`, com senha `poll`. Isso leva aproximadamente cinco minutos para configurar e você pode fazer isso executando o script de SQL mostrado na Listagem 21.3. Isso pode ser feito direcionando script por meio de um login root utilizando:

```
mysql -u root -p < pollsetup.sql
```

Naturalmente, você também poderia utilizar o login de qualquer usuário com os privilégios do MySQL apropriados.

Listagem 21.3 **`pollsetup.sql`** – **Configurando o banco de dados poll**

```
create database poll;
use poll;
create table poll_results (
  candidate varchar(30),
  num_votes int
);
insert into poll_results values
  ('John Smith', 0),
  ('Mary Jones', 0),
  ('Fred Bloggs', 0)
;
grant all privileges
on poll.*
to poll@localhost
identified by 'poll';
```

Esse banco de dados contém três candidatos. Fornecemos uma interface de votação via uma página chamada `vote.html`. O código para essa página é mostrado na Listagem 21.4.

Listagem 21.4 **`vote.html`** – **Usuários podem votar aqui**

```
<html>
<head>
  <title>Polling</title>
<head>
<body>
<h1>Pop Poll</h1>
<p>Who will you vote for in the election?</p>
<form method="post" action="show_poll.php">
<input type="radio" name="vote" value="John Smith">John Smith<br />
<input type="radio" name="vote" value="Mary Jones">Mary Jones<br />
<input type="radio" name="vote" value="Fred Bloggs">Fred Bloggs<br /><br />
<input type="submit"  value="Show results">
</form>
</body>
</html>
```

A saída dessa página é mostrada na Figura 21.7.

A idéia geral é que, quando os usuários clicarem no botão, adicionaremos seu voto ao banco de dados, obteremos todos os votos do banco de dados e desenharemos o gráfico de barras dos resultados atuais.

A típica saída depois que alguns votos foram efetuados é mostrada na Figura 21.8.

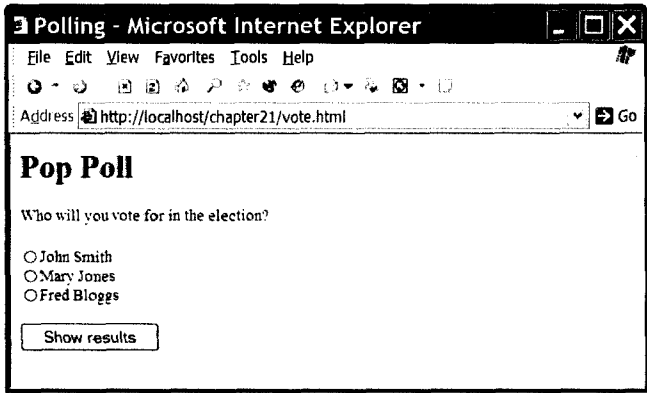



Figura 21.7 Os usuários podem votar aqui e clicar no botão Submit para exibir os resultados atuais da pesquisa.

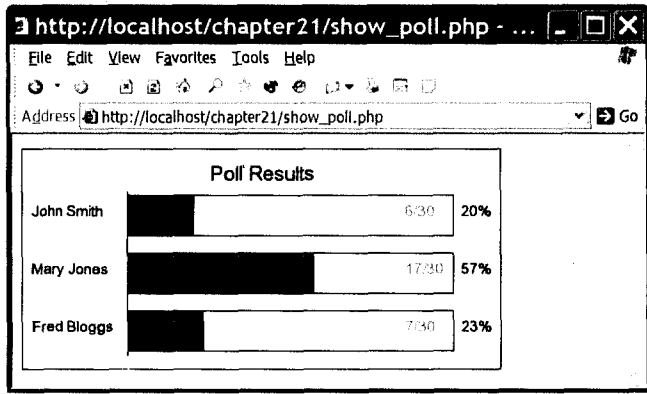



Figura 21.8 Os resultados da pesquisa são criados desenhando uma série de linhas, retângulos e itens de texto em uma tela.

O script que gera essa imagem é bastante longo. Dividimos esse script em quatro partes e discutiremos cada parte separadamente. A maior parte do script é familiar; vimos muitos exemplos do MySQL semelhante a esse. Vimos como pintar um fundo de tela com uma cor sólida e como imprimir rótulos de texto nela.

As novas partes desse script se relacionam a desenhar linhas e retângulos. Focalizaremos nossa atenção nessas seções. A Parte 1 (desse script de quatro partes) é mostrada na Listagem 21.5.1.

Listagem 21.5.1 **show_poll.php – A Parte 1 atualiza o banco de dados de voto e recupera os novos resultados**

```
<?php
/*******************************************
  Consulta de banco de dados para obter informações da enquete
*******************************************/

// obtém voto do formulário
$vote=$HTTP_POST_VARS['vote'];
```

Listagem 21.5.1 Continuação

```
// efetua logon no banco de dados
if (!$db_conn = @mysql_connect('localhost', 'poll', 'poll'))
{
  echo 'Could not connect to db<br />';
  exit;
};
@mysql_select_db('poll');

if (!empty($vote))  // se o usuário preencheu o formulário, adiciona o voto dele
{
  $vote = addslashes($vote);
  $query = "update poll_results
            set num_votes = num_votes + 1
            where candidate = '$vote'";
  if(!($result = @mysql_query($query, $db_conn)))
  {
    echo 'Could not connect to db<br />';
    exit;
  }
};

// obtém resultados atuais da pesquisa eleitoral, independente se eles votaram ou não
$query = 'select * from poll_results';
if(!($result = @mysql_query($query, $db_conn)))
{
  echo 'Could not connect to db<br />';
  exit;
}
$num_candidates = mysql_num_rows($result);

// calcula o número total de votos até agora
$total_votes=0;
while ($row = mysql_fetch_object ($result))
{
    $total_votes +=  $row->num_votes;
}
mysql_data_seek($result, 0);  // reinicializa o ponteiro de resultado
```

A Parte 1, mostrada na Listagem 21.5.1, conecta-se ao banco de dados do MySQL, atualiza os votos de acordo com o que o usuário digitou e obtém os novos votos. Depois que tivermos essas informações, podemos começar a fazer os cálculos para desenhar o gráfico. A Parte 2 é mostrada na Listagem 21.5.2.

Listagem 21.5.2 **showpoll.php** – A Parte 2 configura todas as variáveis para o desenho

```
/*******************************************
  Cálculos iniciais para o gráfico
*******************************************/
// configura constantes
putenv('GD2FONTPATH=C:\WINNT\Fonts');
$width=500;         // largura de imagem em pixels - isso se ajustará em 640x480
$left_margin = 50; // espaço à esquerda da imagem
$right_margin= 50; // idem para a direita
$bar_height = 40;
$bar_spacing = $bar_height/2;
$font = 'arial';
$title_size= 16; // ponto
$main_size= 12; // ponto
```

Listagem 21.5.2 **Continuação**

```
$small_size= 12; // ponto
$text_indent = 10; // posição para rótulos de texto à esquerda

// configura ponto inicial para desenhar a partir de
$x = $left_margin + 60;  // espaço para desenhar linha de base do gráfico
$y = 50;                 // idem
$bar_unit = ($width-($x+$right_margin)) / 100;   // um "ponto" no gráfico

// calcula altura de gráfico - barras mais espaço em branco mais alguma margem
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
```

A Parte 2 configura algumas variáveis que utilizaremos para realmente desenhar o gráfico. Elaborar os valores para esses tipos de variáveis pode ser tedioso, mas um pouco de planejamento sobre a aparência final desejada para a imagem tornará o processo de desenho muito mais fácil. Os valores que utilizamos aqui surgiram esboçando o efeito desejado em um pedaço de papel e estimando as proporções requeridas.

A variável `$width` é a largura total da tela que utilizaremos. Configuraremos também as margens esquerda e direita (com `$left_margin` e `$right_margin`, respectivamente); a "grossura" e o espaçamento entre as barras (`$bar_height` e `$bar_spacing`); e a fonte, os tamanhos de fonte e a posição de rótulo (`$font`, `$title_size`, `$main_size`, `$small_size` e `$text_indent`).

Dados esses valores de base, podemos então fazer alguns cálculos. Queremos desenhar uma linha de base a partir da qual todas as barras se estendem. Podemos elaborar a posição para essa linha de base utilizando a margem esquerda mais um espaço livre para os rótulos de texto para a coordenada X e novamente uma estimativa de nosso esboço para a coordenada Y. Em vez disso, é possível obter a largura exata do nome mais longo se a flexibilidade for importante.

Elaboramos também dois valores importantes: primeiro, a distância no gráfico que representa uma unidade:

```
$bar_unit = ($width-($x+$right_margin)) / 100;   // um "ponto" no gráfico
```

Esse é o comprimento máximo das barras – a partir da linha de base até a margem direita – dividido por 100 porque nosso gráfico mostrará valores de porcentagem.

O segundo valor é a altura total de que precisamos para a tela:

```
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
```

Esse valor é basicamente a altura por barras vezes o número de barras, mais uma quantidade extra para o título. A Parte 3 é mostrada na Listagem 21.5.3.

Listagem 21.5.3 **`showpoll.php` – A Parte 3 configura o gráfico, pronto para os dados serem adicionados**

```
/*******************************************
  Configura imagem de base
*******************************************/
// cria uma tela em branco
$im = imagecreate($width,$height);

// Aloca cores
$white=ImageColorAllocate($im,255,255,255);
$blue=ImageColorAllocate($im,0,64,128);
$black=ImageColorAllocate($im,0,0,0);
$pink = ImageColorAllocate($im,255,78,243);
```

Listagem 21.5.3 **Continuação**

```
$text_color = $black;
$percent_color = $black;
$bg_color = $white;
$line_color = $black;
$bar_color = $blue;
$number_color = $pink;

// Cria a "tela" sobre a qual desenhar
ImageFilledRectangle($im,0,0,$width,$height,$bg_color);

// Desenha um contorno em volta da tela
ImageRectangle($im,0,0,$width-1,$height-1,$line_color);

// Adiciona o título
$title = 'Poll Results';
$title_dimensions = ImageTTFBBox($title_size, 0, $font, $title);
$title_length = $title_dimensions[2] - $title_dimensions[0];
$title_height = abs($title_dimensions[7] - $title_dimensions[1]);
$title_above_line = abs($title_dimensions[7]);
$title_x = ($width-$title_length)/2; // centraliza em x
$title_y = ($y - $title_height)/2 + $title_above_line; // centraliza no intervalo y
ImageTTFText($im, $title_size, 0, $title_x, $title_y,
          $text_color, $font, $title);

// Desenha uma linha de base a partir de um pouco acima da localização
// da primeira barra até um pouco abaixo da última
ImageLine($im, $x, $y-5, $x, $height-15, $line_color);
```

Na Parte 3, configuramos a imagem básica, alocamos as cores e então começamos a desenhar o gráfico.

Preenchemos o fundo para o gráfico dessa vez utilizando:

```
ImageFilledRectangle($im,0,0,$width,$height,$bg_color);
```

A função `ImageFilledRectangle( )`, como você poderia imaginar, desenha um retângulo preenchido. O primeiro parâmetro é, como de costume, o identificador de imagem. Então, devemos passar a ele as coordenadas X e Y do ponto inicial e o ponto terminal do retângulo. Essas coordenadas correspondem aos cantos superior esquerdo e inferior direito, respectivamente. Nesse caso, estamos preenchendo a tela inteira com a cor de fundo, que é o último parâmetro e é branca.

Então chamamos

```
ImageRectangle($im,0,0,$width-1,$height-1,$line_color);
```

para desenhar um contorno preto em torno da borda da tela. Essa função desenha o contorno de retângulo em vez de um retângulo preenchido. Os parâmetros são os mesmos. Observe que desenhamos o retângulo para `$width-1` e `$height-1` – uma tela de largura (width) por altura (height) vai de (0, 0) para esses valores. Se desenhássemos para `$width` e `$height`, o retângulo estaria fora da área de lona.

Utilizamos a mesma lógica e as mesmas funções utilizadas no nosso último script para centralizar e escrever o título no gráfico.

Por fim, desenhamos a linha de base para a barra com:

```
ImageLine($im, $x, $y-5, $x, $height-15, $line_color);
```

A função `ImageLine( )` desenha uma linha na imagem que especificamos (`$im`) a partir de um conjunto de coordenadas (`$x`, `$y-5`) para outro (`$x`, `$height-15`), na cor especificada por `$line_color`.

Nesse caso, desenhamos a linha de base um pouco acima de onde queremos desenhar a primeira barra, um pouco acima da parte inferior da tela.

Agora estamos prontos para preencher os dados no gráfico. A Parte 4 é mostrada na Listagem 21.5.4.

Listagem 21.5.4 **showpoll.php – A Parte 4 desenha os dados reais no gráfico e conclui**

```
/*******************************************
  Desenha dados no gráfico
*******************************************/
// Obtém cada linha de dados do db e desenha barras correspondentes
while ($row = mysql_fetch_object ($result))
{
  if ($total_votes > 0)
    $percent = intval(round(($row->num_votes/$total_votes)*100));
  else
    $percent = 0;

  // exibe porcentagem para esse valor
  ImageTTFText($im, $main_size, 0, $width-30, $y+($bar_height/2),
               $percent_color, $font, $percent.'%');
  if ($total_votes > 0)
   $right_value = intval(round(($row->num_votes/$total_votes)*100));
  else
    $right_value = 0;

  // comprimento da barra para esse valor
  $bar_length = $x + ($right_value * $bar_unit);

  // desenha barra para esse valor
  ImageFilledRectangle($im, $x, $y-2, $bar_length, $y+$bar_height, $bar_color);

  // desenha título para esse valor
  ImageTTFText($im, $main_size, 0, $text_indent, $y+($bar_height/2),
               $text_color, $font, $row->candidate);

  // desenha contorno mostrando 100%
  ImageRectangle($im, $bar_length+1, $y-2,
                ($x+(100*$bar_unit)), $y+$bar_height, $line_color);

  // exibe números
  ImageTTFText($im, $small_size, 0, $x+(100*$bar_unit)-50, $y+($bar_height/2),
               $number_color, $font, $row->num_votes.'/'.$total_votes);

  // move-se para baixo até a próxima barra
  $y=$y+($bar_height+$bar_spacing);
}

/*******************************************
  Exibe a imagem
*******************************************/
Header('Content-type:  image/png');
ImagePng($im);


/*******************************************
  Limpa
*******************************************/
ImageDestroy($im);
?>
```

A Parte 4 passa um por um os candidatos do banco de dados, elabora a porcentagem de votos e desenha as barras e rótulos para cada candidato.

Novamente adicionamos rótulos utilizando ImageTTFText( ). Desenhamos as barras como retângulos preenchidos utilizando ImageFilledRectangle( ):

```
ImageFilledRectangle($im, $x, $y-2, $bar_length, $y+$bar_height, $bar_color);
```

Adicionamos contornos à marca de 100% utilizando ImageRectangle( ):

```
ImageRectangle($im, $bar_length+1, $y-2,
            ($x+(100*$bar_unit)), $y+$bar_height, $line_color);
```

Depois de desenharmos todas as barras, novamente enviamos a imagem para a saída utilizando ImagePNG( ) e limpamos tudo utilizando ImageDestroy( ).

Esse é um script longo, mas pode ser facilmente adaptado para atender às suas necessidades ou para autogerar pesquisas eleitorais por meio de uma interface. Um recurso importante que está faltando nesse script é qualquer tipo de mecanismo antifraude. Os usuários descobririam rapidamente que podem votar repetidamente e tornar o resultado sem sentido.

Você pode utilizar uma abordagem semelhante para desenhar gráficos de linha e até gráficos de pizza, se for bom em matemática.

Usando outras funções de imagem

Além das funções de imagem que utilizamos neste capítulo, há funções para permitir desenhar linhas curvas (ImageArc( )) e polígonos (ImagePolygon( )), bem como variações das formas que utilizamos aqui. Sempre comece esboçando o que deseja desenhar e então você pode buscar no manual quaisquer funções extras de que possa precisar.

Leitura adicional

Uma grande quantidade de material de leitura está disponível on-line. Se você estiver tendo problemas com as funções de imagem, às vezes ajuda ver a documentação de fonte para o GD porque as funções de PHP são empacotadores para essa biblioteca. A documentação GD está disponível em:

http://www.boutell.com/gd/

Lembre-se também de que a versão PHP de GD2 é uma bifurcação da biblioteca principal, por isso alguns detalhes podem variar.

Há também alguns tutoriais excelentes sobre os tipos de aplicaçõeos particulares de gráfico, particularmente em Zend e Devshed:

http://www.zend.com

http://devshed.com

A aplicação de gráfico de barras neste capítulo foi inspirada no script de gráfico de barras dinâmico escrito por Steve Maranda, disponível em Devshed.

A seguir

No próximo capítulo, abordaremos a útil funcionalidade de controle de sessão do PHP.

22

Utilizando controle de sessão no PHP

ESTE CAPÍTULO DISCUTIRÁ A FUNCIONALIDADE de controle de sessão no PHP.

Os tópicos-chave deste capítulo são:

- O que é controle de sessão;
- Cookies;
- Configurando uma sessão;
- Variáveis de sessão;
- Sessões e autenticação.

O que é controle de sessão

Talvez você tenha ouvido que "HTTP é um protocolo sem informações de estado". Isso significa que o protocolo não tem nenhuma maneira predefinida de manter o estado entre duas transações. Quando um usuário solicita uma página, seguida de outra, o HTTP não fornece uma maneira de dizer se as duas solicitações vêm do mesmo usuário.

A idéia de controle de sessão é ser capaz de monitorar um usuário durante uma única sessão em um Web site. Se pudermos fazer isso, podemos facilmente suportar o login de um usuário e mostrar o conteúdo de acordo com seu nível de autorização ou preferências pessoais. Podemos monitorar o comportamento do usuário e implementar carrinhos de compras.

Desde a versão 4, o PHP inclui funções de controle de sessão nativas. A abordagem para controle de sessão mudou um pouco com a introdução das variáveis superglobais; a superglobal `@_SESSION` está disponível para uso agora.

Entendendo a funcionalidade de sessão básica

As sessões no PHP têm por base um ID único de sessão, um número criptograficamente aleatório. Esse ID de sessão é gerado pelo PHP e armazenado no lado cliente pelo tempo de vida de uma sessão. Ele pode ser armazenado no computador de um usuário em um cookie ou passado junto com URLs.

O ID de sessão atua como uma chave que permite registrar variáveis particulares como as chamadas variáveis de sessão. O conteúdo dessas variáveis é armazenado no servidor. O ID de sessão é a única informação visível no lado cliente. Se, no momento de uma conexão particular com seu site, o ID de sessão for visível por um cookie ou pelo URL, você pode acessar as variáveis de sessão armazenadas no servidor para essa sessão. Por padrão, as variáveis de sessão são armazenadas em arquivos simples no servidor. (Você pode alterar isso para utilizar um banco de dados se estiver disposto a escrever sua própria função – mais sobre isso na seção "Configurando o controle de sessão".)

Provavelmente, você utilizou Web sites que armazenam um ID de sessão no URL. Se houver uma string de dados de aparência aleatória no seu URL, é possível que ela seja alguma forma de controle de sessão.

Os cookies são uma solução diferente para o problema de preservar estado entre várias transações enquanto ainda mantém um URL de aparência limpa.

O que é um cookie?

Um *cookie* é uma pequena parte das informações que os scripts podem armazenar em uma máquina do lado cliente. Você pode configurar um cookie em uma máquina do usuário enviando um cabeçalho de HTTP contendo dados no seguinte formato:

```
Set-Cookie: NOME=VALOR ; [expires=DATA;] [path=CAMINHO;]
 [domain=NOME_DE_DOMÍNIO;] [secure]
```

Isso criará um cookie chamado `NOME` com o valor `VALOR`. Todos os outros parâmetros são opcionais. O campo `expires` configura uma data além da qual o cookie não é mais relevante. (Observe que se nenhuma data de expiração for configurada, o cookie será efetivamente permanente a menos que manualmente excluído por você ou pelo usuário.) Juntos, o caminho e o domínio podem ser utilizados para especificar o URL ou URLs para os quais o cookie é relevante. A palavra-chave `secure` significa que o cookie não será enviado em uma conexão de HTTP simples.

Quando um navegador conecta-se a um URL, primeiro ele pesquisa os cookies armazenados localmente. Se algum deles for relevante para o URL sendo conectado, eles serão transmitidos de volta para o servidor.

Configurando cookies de PHP

Você pode configurar manualmente os cookies no PHP utilizando a função `setcookie( )`. Ela tem o seguinte protótipo:

```
bool setcookie (string nome [, string valor [, int data_de_expiração [, string caminho
[, string domínio [, int seguro]]]]])
```

Os parâmetros correspondem exatamente aos parâmetros no cabeçalho `Set-Cookie` mencionado anteriormente.

Se configurar um cookie como

```
setcookie ('mycookie', 'value');
```

quando o usuário visitar a próxima página do seu site (ou recarregar a página atual), você terá acesso ao cookie por meio de `$_COOKIE['mycookie']` ou `$HTTP_COOKIE_VARS["mycookie"]`. (Ou, se você tiver `register_globals` ativado, diretamente como `$mycookie`.)

Você pode excluir um cookie chamando `setcookie( )` novamente com o mesmo nome do cookie e com uma data/hora de expiração no passado. Você também pode configurar um cookie manualmente via função `Header( )` e a sintaxe de cookie dada anteriormente. Uma dica é que os cabeçalhos de cookie devem ser enviados *antes de qualquer outro cabeçalho* ou eles não funcionarão. (Isso é uma limitação de cookie, em vez de uma limitação do PHP.)

Utilizando cookies com sessões

Os cookies têm algum problemas associados: alguns navegadores não aceitam cookies e talvez alguns usuários tenham desativado os cookies em seus navegadores. Essa é uma das razões de as sessões de PHP utilizarem um método cookie/URL dual. (Discutiremos mais sobre isso em um minuto.)

Quando estiver utilizando sessões de PHP, você não terá de configurar os cookies manualmente. As funções de sessão cuidarão disso para você.

Você pode utilizar a função `session_get_cookie_params( )` para examinar o conteúdo do cookie configurado pelo controle de sessão. Isso retorna um array associativo contendo os elementos `lifetime`, `path`, `domain` e `secure`.

Você também pode utilizar

```
session_set_cookie_params($tempoDeVida, $caminho, $domínio [, $seguro]);
```

para configurar os parâmetros de sessão do cookie.

Se quiser ler mais sobre os cookies, você pode consultar a especificação de cookie no site do Netscape:

http://home.netscape.com/newsref/std/cookie_spec.html

(Você provavelmente pode ignorar o fato de que esse documento chama a si próprio de uma "especificação preliminar" – tem sido dessa maneira desde 1995.)

Armazenando o id da sessão

Por padrão, o PHP utilizará os cookies com sessões. Se possível, um cookie será configurado para armazenar o ID da sessão.

O outro método que ele pode utilizar é adicionar o ID da sessão ao URL. Você pode configurar para que isso aconteça automaticamente se configurar a diretiva `session.use_trans_sid` no arquivo `php.ini`. Fica desligado por padrão.

Como alternativa, você pode embutir manualmente o ID da sessão em links de modo que ele seja passado. O ID da sessão é armazenado na constante SID. Para passá-lo manualmente, você o adiciona ao fim de um link semelhante a um parâmetro `GET`:

```
<A HREF="link.php?<?php echo strip_tags(SID); ?>">
```

(A função `strip_tags` é usada aqui para evitar ataques de script entre sites.)

Geralmente, é mais fácil compilar com `--enable-trans-sid`, onde possível.

Implementando sessões simples

Os passos básicos para utilizar as sessões são:

- Iniciar uma sessão;
- Registrar variáveis de sessão;
- Utilizar variáveis de sessão;
- Remover registros de variáveis e destruir a sessão.

Observe que todos esses passos não acontecem necessariamente no mesmo script e alguns deles acontecerão em vários scripts. Vamos conversar sobre cada um desses passos de cada vez.

Iniciando uma sessão

Antes de utilizar a funcionalidade da sessão, você precisa realmente iniciar uma sessão. Há três maneiras de fazer isso.

A primeira, e a mais simples, é iniciar um script com uma chamada para a função `session_start( )`:

```
session_start( );
```

Essa função verifica se já há um ID de sessão atual. Se não houver, essa função criará um. Se já existir um, ele essencialmente carrega as variáveis registradas de sessão para que você possa utilizá-las.

É uma boa idéia chamar `session_start( )` no início de todos os seus scripts que utilizam sessões.

A segunda maneira de começar uma sessão é configurar o PHP para iniciar uma automaticamente quando alguém vier para seu site. Você pode fazer isso com a opção `session.auto_start` no seu arquivo `php.ini` – veremos isso quando discutirmos a configuração.

Registrando variáveis de sessão

O registro de variáveis de sessão foi recentemente alterado no PHP. As variáveis de sessão são armazenadas no array superglobal `$_SESSION` desde o PHP 4.1 e também no `$HTTP_SESSION_VARS` mais antigo. Recomendamos usar `$_SESSION`. A fim de criar uma variável de sessão, você simplesmente configura um elemento em um desses arrays, como a seguir:

```
$_SESSION['myvar'] = 5;
```

Se você está utilizando uma versão mais antiga do PHP para que uma variável seja monitorada de um script para outro, poderá registrá-la com uma chamada a `session_register( )`. Isso agora está em desuso.

A variável de sessão que você acabou de criar será monitorada até que a sessão seja finalizada ou até que seja desconfigurada manualmente.

Utilizando variáveis de sessão

Para colocar as variáveis de sessão em escopo de modo que possam ser utilizadas, primeiro inicie uma sessão utilizando `session_start( )`. Então, você pode acessar a variável via array superglobal `$_SESSION` – por exemplo, como `$_SESSION['myvar']`.

Ao utilizar um objeto como variável de sessão, é importante incluir a definição de classe antes de chamar `session_start( )` para recarregar as variáveis de sessão. Dessa forma, o PHP sabe como reconstruir o objeto da sessão.

Se tiver `register_globals` ativado, você pode acessar as variáveis de sessão usando seus nomes no formato abreviado – por exemplo, `$myvar` – mas essa abordagem não é recomendada. Se tiver `register_globals` ativado, ao codificar, lembre-se de que uma variável de sessão não pode ser sobrescrita por dados `GET` ou `POST`, o que é um bom recurso de segurança.

Por outro lado, você precisa ser cuidadoso ao verificar se as variáveis de sessão foram configuradas (via, digamos, `isset( )` ou `empty( )`). Lembre-se de que as variáveis podem ser configuradas pelo usuário via `GET` ou `POST`. Você pode ver se ela é uma variável de sessão registrada verificando em `$_SESSION`.

É possível verificar isso diretamente utilizando o seguinte exemplo:

```
if (isset($_SESSION['myvar'])) ...
```

Removendo registro de variáveis e destruindo a sessão

Quando tiver terminado de usar uma variável de sessão, você pode desconfigurá-la. Faça isso diretamente desconfigurando o elemento apropriado do array `$_SESSION`, como neste exemplo:

```
unset($_SESSION['myvar']);
```

Observe que o uso de `session_unregister( )` e `session_unset( )` não é mais requerido, nem recomendado. Essas funções eram utilizadas antes da introdução de `$_SESSION`.

Você não deveria tentar desconfigurar o array completo `$_SESSION` porque isso na verdade desativará sessões. Para desconfigurar todas as variáveis de sessão de uma vez, use:

```
$_SESSION = array( );
```

Quando tiver terminado uma sessão, você deve primeiro remover o registro de todas as variáveis e então chamar

```
session_destroy( );
```

para limpar o ID de sessão.

Criando um exemplo de sessão simples

Uma parte dessa discussão pode parecer um pouco abstrata, então vejamos um exemplo. Vamos implementar um conjunto de três páginas.

Na primeira página, iniciaremos uma sessão e registraremos a variável `$HTTP_SESSION_VARS['sess_var']`. O código para fazer isso é mostrado na Listagem 22.1.

Listagem 22.1 **page1.php – Iniciando uma sessão e registrando uma variável**

```
<?php
  session_start( );

  $HTTP_SESSION_VARS['sess_var'] = "Hello world!";

  echo 'The content of $HTTP_SESSION_VARS[\'sess_var\'] is '
        .$HTTP_SESSION_VARS['sess_var'].'<br />';
?>
<a href="page2.php">Next page</a>
```

Registramos a variável e configuramos seu valor. A saída desse script é mostrada na Figura 22.1.

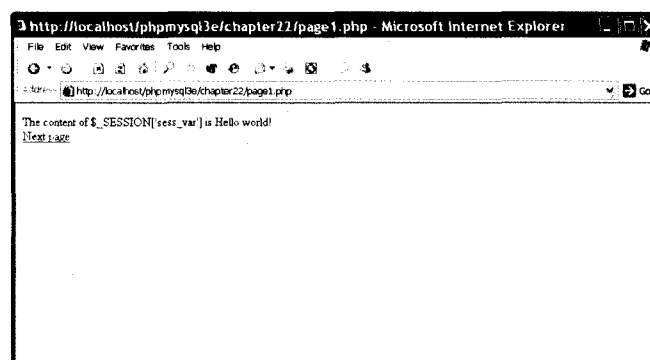



Figura 22.1 O valor inicial da variável da sessão mostrado pela `page1.php`.

O valor *final* da variável na página é o valor que estará disponível nas páginas subseqüentes. No final do script, a variável de sessão é *serializada*, ou congelada, até ser recarregada via próxima chamada para `session_start( )`.

Portanto, começamos o próximo script chamando `session_start( )`. Esse script é mostrado na Listagem 22.2.

Listagem 22.2 **page2.php – Acessando uma variável de sessão e removendo-a**

```
<?php
  session_start( );

  echo 'The content of $HTTP_SESSION_VARS[\'sess_var\'] is '
        .$HTTP_SESSION_VARS['sess_var'].'<br />';

  unset($HTTP_SESSION_VARS['sess_var']);
?>
<a href="page3.php">Next page</a>
```

Depois de chamar session_start( ), a variável $HTTP_SESSION_VARS['sess_var'] estará disponível com seu valor previamente armazenado, como você pode ver na Figura 22.2.

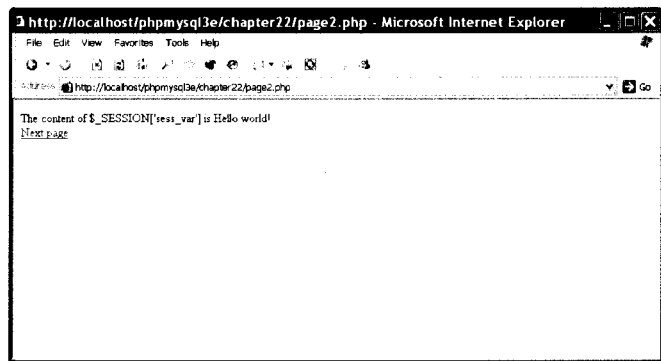



Figura 22.2 O valor da variável de sessão foi passada via ID de sessão para page2.php.

Depois que utilizamos a variável, a removemos. A sessão ainda existe, mas a variável $HTTP_SESSION_VARS['sess_var'] não existe mais.

Por fim, passamos para a page3.php, o script final em nosso exemplo. O código para esse script é mostrado na Listagem 22.3.

Listagem 22.3 **page3.php – Terminando a sessão**

```
<?php

  session_start( );
 echo 'The content of $_SESSION[\'sess_var\'] is '
       .$_SESSION['sess_var'].'<br />';

  session_destroy( );
?>
```

Como você pode ver na Figura 22.3, não temos mais acesso ao valor persistente de $_SESSION['sess_var'].

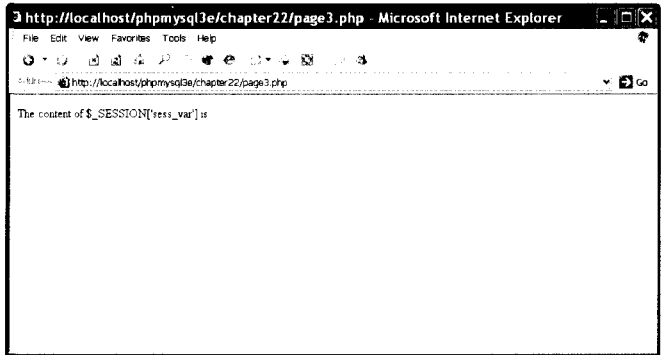



Figura 22.3 A variável com o registro removido não está mais disponível.

Com algumas versões do PHP anteriores à 4.3, você talvez encontre um bug ao tentar remover o registro de elementos de $HTTP_SESSION_VARS ou $_SESSION. Se descobrir que não é capaz de remover o registro de elementos (isto é, eles permanecem configurados) você poderá reverter utilizando session_unregister( ) para limpar essas variáveis.

Concluímos chamando session_destroy( ) para determinar o ID de sessão.

Configurando o controle de sessão

Há um conjunto de opções de configuração para as sessões que você pode configurar no seu arquivo php.ini. Algumas opções mais úteis e uma descrição de cada uma são mostradas na Tabela 22.1.

Tabela 22.1 **Opções de configuração de sessão**

Nome da opção	Padrão	Efeito
session.auto_start	0 (desativado)	Inicia sessões automaticamente.
session.cache_expire	180	Configura o tempo de vida para páginas de sessão armazenadas em cache, em minutos.
session.cookie_domain	nenhum	Domínio para configurar no cookie da sessão.
session.cookie_lifetime	0	Quanto tempo o cookie com o ID da sessão vai durar na máquina do usuário. O padrão, 0, é durar até que o navegador seja fechado.
session.cookie_path	/	Caminho para configurar cookie de sessão.
session.name	PHPSESSID	Nome da sessão que é utilizado como o nome de cookie em um sistema do usuário.
session.save_handler	arquivos	Define onde dados de sessão são armazenados. Você pode configurar isso apontando para um banco de dados, mas tem de escrever suas próprias funções.
session.save_path	/tmp	O caminho em que dados de sessão são armazenados. Em geral, o argumento passado para o save tratado e definido pelo session.save_handler.
session.use_cookies	1 (ativado)	Configura sessões para utilizar cookies no lado de cliente.

Implementando autenticação com controle de sessão

Por fim, examinaremos um exemplo mais substancial utilizando controle de sessão.

Possivelmente, a utilização mais comum de controle de sessão seja monitorar usuários depois que foram autenticados via um mecanismo de login. Nesse exemplo, combinaremos autenticação a partir de um banco de dados do MySQL com o uso de sessões para fornecer essa funcionalidade. Essa funcionalidade formará a base do projeto no Capítulo 26, e será reutilizada em outros projetos. Reutilizaremos a autenticação do banco de dados que configuramos no Capítulo 16, para utilizar mod_auth_mysql. Você pode verificar a Listagem 16.3 nesse capítulo para detalhes do banco de dados.

O exemplo consiste em três scripts simples. O primeiro, authmain.php, fornece um formulário de login e autenticação para membros de nosso Web site. O segundo, members_only.php, exibe informações somente para membros que efetuaram logon com sucesso. O terceiro, logout.php, efetua logout de um membro.

Para entender como isso funciona, veja a Figura 22.4. Essa é a página inicial exibida por authmain.php.

Essa página fornece ao usuário um lugar para efetuar login. Se o usuário tentar acessar a seção Members sem efetuar login primeiro, ele obterá a mensagem mostrada na Figura 22.5.

Entretanto, se o usuário efetuar login primeiro (com nome de usuário: testuser e senha: test123 como configurado no Capítulo 16) e então tentar ver a página Members, ele obterá a saída mostrada na Figura 22.6.

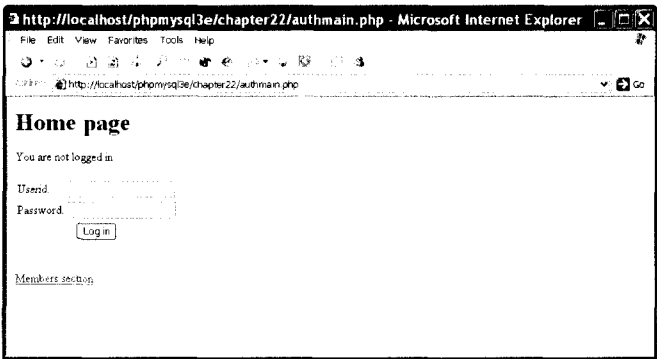



Figura 22.4 Como o usuário ainda não efetuou logon, mostra a ele uma página de login.

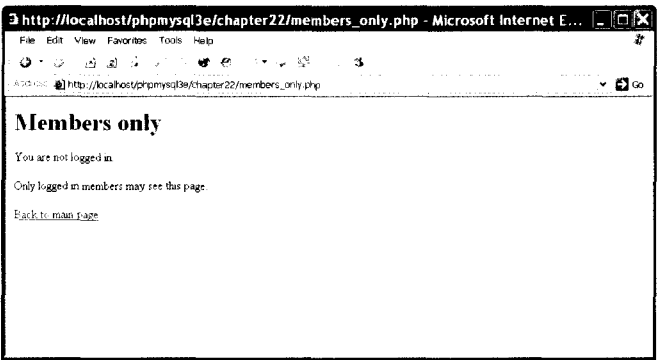



Figura 22.5 Os usuários que não efetuaram logon não podem ver o conteúdo de site; em vez disso será mostrada essa mensagem a eles.

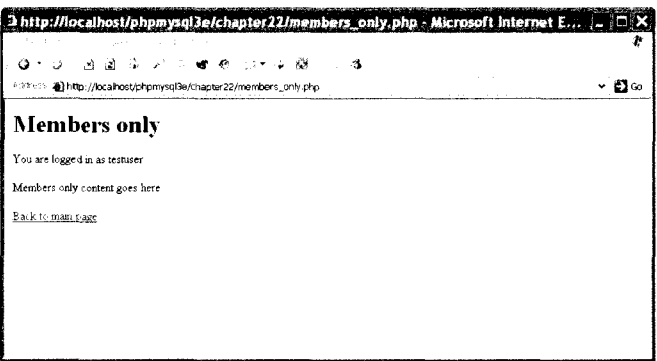



Figura 22.6 Depois que o usuário efetuou logon, ele pode acessar as áreas Members.

Vejamos o código para essa aplicação. A maior parte do código está em `authmain.php`. Esse script pode ser visto na Listagem 22.4. Nós o examinaremos aqui detalhadamente.

Listagem 22.4 **authmain.php – Parte principal da aplicação de autenticação**

```
<?php
session_start( );

if (isset($_POST['userid']) && isset($_POST['password']))
{
  // se o usuário acabou de tentar efetuar login
  $userid = $_POST['userid'];
  $password = $_POST['password'];
```

Listagem 22.4 **Continuação**

```
  $db_conn = mysql_connect('localhost', 'webauth', 'webauth');
  mysql_select_db('auth', $db_conn);
  $query = 'select * from auth '
          ."where name='$userid' "
          ." and pass=password('$password')";
  $result = mysql_query($query, $db_conn);
  if (mysql_num_rows($result) >0 )
  {
 // se o usuário estiver no banco de dados, registra o id do usuário
    $_SESSION['valid_user'] = $userid;
  }
}
?>
<html>
<body>
<h1>Home page</h1>
<?
  if (isset($_SESSION['valid_user']))
  {
    echo 'You are logged in as: '.$_SESSION['valid_user'].' <br />';
    echo '<a href="logout.php">Log out</a><br />';
  }
  else
  {
    if (isset($userid))
    {
      // se o usuário tentar efetuar login e falhar
      echo 'Could not log you in';
    }
    else
    {
      // o usuário não tentou efetuar login ainda ou efetuou logout
      echo 'You are not logged in.<br />';
    }

    // fornece um formulário para efetuar login
    echo '<form method="post" action="authmain.php">';
    echo '<table>';
    echo '<tr><td>Userid:</td>';
    echo '<td><input type="text" name="userid"></td></tr>';
    echo '<tr><td>Password:</td>';
    echo '<td><input type="password" name="password"></td></tr>';
    echo '<tr><td colspan="2" align="center">';
    echo '<input type="submit" value="Log in"></td></tr>';
    echo '</table></form>';
  }
?>
<br>
<a href="members_only.php">Members section</a>
</body>
</html>
```

A lógica desse script é relativamente complicada porque ele simultaneamente: exibe o formulário de login, que é a ação do formulário, e contém HTML para uma tentativa malsucedida ou bem-sucedida de login.

As atividades do script giram em torno da variável de sessão `$valid_user`. A idéia básica é que se alguém efetuar login com sucesso, registraremos uma variável de sessão chamada `$_SESSION['valid_user']` que contém seu userid.

A primeira coisa que fazemos no script é chamar session_start( ). Isso carregará na variável de sessão valid_user se ela tiver sido registrada.

Na primeira passagem pelo script, nenhuma das condições if se aplicará e o usuário parará no final do script, onde o informaremos que ele não está conectado e forneceremos uma forma de fazer isso:

```
echo '<form method="post" action="authmain.php">';
echo '<table>';
echo '<tr><td>Userid:</td>';
echo '<td><input type="text" name="userid"></td></tr>';
echo '<tr><td>Password:</td>';
echo '<td><input type="password" name="password"></td></tr>';
echo '<tr><td colspan="2" align="center">';
echo '<input type="submit" value="Log in"></td></tr>';
echo '</table></form>';
```

Quando ele pressionar o botão submit no formulário, esse script é reinvocado e iniciamos novamente a partir da parte superior. Dessa vez, teremos um userid e uma senha para autenticar, armazenados como $_POST['userid'] e $_POST['password']. Se essas variáveis estiverem configuradas, entramos no bloco de autenticação:

```
if (isset($_POST['userid']) && isset($_POST['password']))
{
  // se o usuário acabou de tentar efetuar login
  $userid = $_POST['userid'];
  $password = $_POST['password'];

  $db_conn = new mysqli_connect('localhost', 'webauth', 'webauth', 'auth');

if (mysqli_connect_errno( )) {
echo 'Connection to database failed: ' .mysqli_connect_error( );
exit( );
}

$query = 'select * from authorized_users '
        ."where name= '$userid' "
        ." and password=sha1('password')";

$result = $db_conn->query($query);
```

Conectamos-nos a um banco de dados do MySQL e verificamos o userid e a senha. Se esses forem um par correspondente no banco de dados, criamos a variável $_SESSION['valid_user'] que contém o userid para esse usuário particular, então saberemos quem está conectado mais adiante no monitoramento.

```
  if ($result->num_rows >0 )
  {
    // se eles estiverem no banco de dados, registra o id do usuário
    $_SESSION['valid_user'] = $userid;
  }
 $db_conn->close( );
}
```

Como sabemos agora quem é o usuário, não precisamos mostrar a ele o formulário de login outra vez. Em vez disso, diremos que sabemos quem ele é e forneceremos a opção de efetuar logout:

```
if (isset($_SESSION['valid_user']))
{
  echo 'You are logged in as: '.$_SESSION['valid_user'].' <br />';
  echo '<a href="logout.php">Log out</a><br />';
}
```

Se tentarmos efetuar login do usuário e falharmos por alguma razão, teremos um userid mas não uma variável `$_SESSION['valid_user']`, então podemos apresentar uma mensagem de erro:

```
if (isset($userid))
{
  // se o usuário tentar efetuar login e falhar
  echo 'Could not log you in.<br />';
}
```

Isso é tudo quanto ao script principal. Agora, vejamos a página Members. O código para esse script é mostrado na Listagem 22.5.

Listagem 22.5 **`members_only.php` – O código para a seção dos membros de nosso Web site verifica usuários válidos**

```
<?php
  session_start( );

  echo '<h1>Members only</h1>';

  // verifica a variável de sessão

  if (isset($_SESSION['valid_user']))
  {
    echo '<p>You are logged in as '.$_SESSION['valid_user'].'</p>';
    echo '<p>Members only content goes here</p>';
  }
  else
  {
    echo '<p>You are not logged in.</p>';
    echo '<p>Only logged in members may see this page.</p>';
  }

  echo '<a href="authmain.php">Back to main page</a>';
?>
```

Esse código é muito simples. Tudo que faz é iniciar uma sessão e examinar se a sessão atual contém um usuário registrado verificando se o valor de `$_SESSION['valid_user']` está configurado. Se o usuário tiver efetuado login, mostramos o conteúdo de Members; caso contrário, informamos que ele não é autorizado.

Por fim temos o script `logout.php` que desconecta um usuário do sistema. O código para esse script é mostrado na Listagem 22.6.

Listagem 22.6 **`logout.php` – Esse script remove o registro da variável de sessão e destrói a sessão**

```
<?php
  session_start( );

  // armazena para testar se eles *estavam* conectados
 $old_user = $_SESSION['valid_user'];
  unset($_SESSION['valid_user']);
  session_destroy( );
?>
<html>
<body>
<h1>Log out</h1>
<?php
  if (!empty($old_user))
```

Listagem 22.6 **Continuação**

```
  {
    echo 'Logged out.<br />';
  }
  else
  {
    // se eles não efetuaram logon mas vieram para esta página de alguma maneira
    echo 'You were not logged in, and so have not been logged out.<br />';
  }
?>
<a href="authmain.php">Back to main page</a>
</body>
</html>
```

O código é muito simples, mas é necessário fazer alguns aprimoramentos. Iniciamos uma sessão, armazenamos o nome de usuário antigo do usuário, removemos o registro da variável valid_user e destruímos a sessão. Em seguida, fornecemos uma mensagem que será diferente para o usuário se ele efetuou o logout ou se não estava conectado.

Esse conjunto simples de scripts formará a base para uma grande quantidade de trabalho que faremos nos capítulos posteriores.

Leitura adicional

Você pode ler mais sobre cookies em: http://home.netscape.com/newsref/std/cookie_spec.html

A seguir

Quase concluímos esta seção do livro.

Antes de mudarmos para os projetos, discutiremos brevemente alguns outros recursos úteis no PHP que não abordamos em outra parte.

23

Outros recursos úteis

ALGUMAS FUNÇÕES E RECURSOS DO PHP úteis não se encaixam em qualquer categoria particular. Este capítulo explicará esses recursos.

Os tópicos-chave deste capítulo incluem:

- Utilizando aspas mágicas;
- Avaliando strings com `eval( )`;
- Concluindo a execução: `die` e `exit`;
- Serializando variávies e objetos;
- Obtendo informações sobre o ambiente de PHP;
- Alterando temporariamente o ambiente de tempo de execução;
- Carregando extensões do PHP;
- Destacando código-fonte;
- Utilizando o PHP na linha de comando.

Utilizando aspas mágicas

Você provavelmente notou que precisa ter cuidado ao utilizar símbolos de pontuação (' e ") e barras invertidas (\) dentro de strings. O PHP se confundirá com a tentativa de emitir uma instrução de string como esta:

```
echo "color = "#FFFFFF"";
```

e fornecerá um erro de análise sintática. Para incluir aspas dentro de uma string, utilize o tipo de aspas que seja diferente das aspas que incluem a string. Por exemplo:

```
echo "color = '#FFFFFF'";
```

ou

```
echo 'color = "#FFFFFF"';
```

são válidos.

O mesmo problema ocorre com entrada de usuário, bem como entrada e saída para ou a partir de outros programas.

Tentar executar uma consulta `mysql` como:

```
insert into company values ('Bob's Auto Parts');
```

confundirá de modo semelhante o analisador de sintaxe do MySQL.

Já vimos o uso de addslashes( ) e stripslashes( ), que associarão caracteres de escape a quaisquer aspas simples, aspas duplas, barras invertidas e caracteres NULL.

O PHP tem a capacidade útil de automática ou "magicamente" adicionar e remover barras para você. Com duas configurações no arquivo php.ini, você pode ativar ou desativar aspas mágicas para GET, POST, dados de cookie e outras origens.

O valor da diretiva magic_quotes_gpc controla se aspas mágicas são utilizadas para operações GET, POST e de cookie.

Com magic_quotes_gpc ativada, se alguém digitou "Bob's Auto Parts" em um formulário no seu site, seu script receberia "Bob\'s Auto Parts" porque as aspas serão "escapadas" para você. Esse comportamento é muito útil, mas precisamos saber que está acontecendo, então lembre-se de remover as barras antes de ecoar os dados de volta para seus usuários. Isso é fácil se o código executa em um servidor, mas se você estiver escrevendo código para distribuir, vai querer fazê-lo funcionar com ou sem aspas mágicas.

A função get_magic_quotes_gpc( ) retorna 1 ou 0, informando o valor atual de magic_quotes_gpc. Isso é mais útil para testar se você precisa de stripslashes( ) nos dados recebidos do usuário.

O valor de magic_quotes_runtime controla se aspas mágicas são utilizadas por funções que obtêm dados a partir de bancos de dados e arquivos. Para obter o valor de magic_quotes_runtime, utilize a função get_magic_quotes_runtime( ). Essa função retorna 1 ou 0. As aspas mágicas podem ser ativadas para um script particular usando a função set_magic_quotes_runtime( ).

Por default, magic_quotes_gpc está ativado e magic_quotes_runtime está desativado.

Avaliando strings: eval( )

A função eval( ) avaliará uma string como um código de PHP. Por exemplo,

```
eval ( "echo 'Hello World';" );
```

obterá o conteúdo da string e o executará. Essa linha produzirá a mesma saída que:

```
echo 'Hello World';
```

Há uma variedade de casos em que eval( ) pode ser útil. Talvez você queira armazenar blocos de código em um banco de dados e recuperar e avaliá-los com eval( ) em um ponto adiante. Talvez você queira gerar código em um loop e então utilizar eval( ) para executá-lo.

O uso mais comum para eval( ) é como parte de um sistema de amostras. Podemos carregar uma mistura de HTML, PHP e texto simples de um banco de dados. O sistema de amostras pode formatar esse conteúdo e depois executá-lo usando eval para executar qualquer código PHP.

Você pode utilizar eval( ) de modo útil para atualizar ou corrigir código existente. Se tivesse uma coleção grande de scripts que precisasse de uma alteração previsível, seria possível (mas ineficiente) escrever um script para carregar um script antigo em uma string, executar um regexp para fazer alterações e então utilizar eval( ) para executar o script modificado.

É ainda concebível que uma pessoa muito confiável em algum lugar talvez queira permitir que o código de PHP seja inserido em um navegador e executado no servidor dela.

Terminando a execução: die e exit

Até agora neste livro utilizamos a construção de linguagem exit para interromper a execução de um script. Como você provavelmente lembra, essa construção aparece em uma linha sozinha, assim:

```
exit;
```

e não retorna nada. Você pode usar alternativamente o alias die( ).

Para uma terminação ligeiramente mais útil, podemos passar um parâmetro para `exit( )`. Isso pode ser utilizado para enviar para a saída uma mensagem de erro ou executar uma função antes de terminar um script. Isso será familiar aos programadores de Perl. Por exemplo:

```
exit('Script ending now');
```

Mais comumente ela é combinada com um `or` para uma instrução que talvez falhe, como abrir um arquivo ou conectar-se a um banco de dados:

```
mysql_query($query) or die('Could not execute query');
```

Em vez de simplesmente imprimir uma mensagem de erro, você pode chamar uma última função antes de o script terminar:

```
function err_msg( )
{
    echo 'MySQL error was: ';
    echo mysql_error( );
}

mysql_query($query) or die(err_msg( ));
```

Essa abordagem pode ser útil como uma forma de oferecer ao usuário algum motivo por que o script falhou ou como uma forma de fechar elementos HTML ou limpar uma página incompleta do buffer de saída.

Você também pode enviar um e-mail a si mesmo para saber se um erro maior ocorreu ou também pode adicionar erros a um arquivo de log ou lançar uma exceção.

Serializando variáveis e objetos

A serialização é o processo de transformar o que você pode armazenar em uma variável ou objeto do PHP em um fluxo de bytes que possa ser armazenado em um banco de dados ou passado via um URL de página para página. Sem isso, é difícil armazenar ou passar o conteúdo inteiro de um array ou objeto.

A serialização perdeu um pouco de sua utilidade desde a introdução do controle de sessão. A serialização de dados era principalmente utilizada para fazer o que agora é feito com o controle de sessão. De fato, as funções de controle de sessão serializam variáveis de sessão para que elas sejam armazenadas entre solicitações de HTTP.

Mas talvez você ainda queira armazenar um array de PHP ou objeto em um arquivo ou banco de dados. Se esse for o caso, há duas funções que você precisa saber utilizar: `serialize( )` e `unserialize( )`.

Você pode chamar a função `serialize( )` da seguinte maneira:

```
$serial_object = serialize($my_object);
```

Se quiser saber o que a serialização realmente faz, examine o que é retornado dela. Ela converte o conteúdo de um objeto ou array em uma string.

Por exemplo, podemos ver a saída da execução de `serialize( )` sobre um objeto employee simples, definido e instanciado assim:

```
class employee
{
  var $name;
  var $employee_id;
};

$this_emp = new employee;
$this_emp->name = 'Fred';
$this_emp->employee_id = 5324;
```

Se serializarmos isso e o ecoarmos para o navegador, a saída será:

```
O:8:"employee":2:{s:4:"name";s:4:"Fred";s:11:"employee_id";i:5324;}
```

É relativamente fácil ver o relacionamento entre os dados originais do objeto e os dados serializados.

Como os dados serializados são simplesmente textos, você pode gravá-los em um banco de dados ou onde quiser. Esteja ciente de que você deve adicionar barras com addslashes( ) a quaisquer dados antes de gravá-los em um banco de dados, como de costume. Você pode ver a necessidade disso atentando para as aspas na string serializada anterior.

Para obter o objeto de volta, chame unserialize( ):

```
$new_object = unserialize($serial_object);
```

Obviamente, se você chamou addslashes( ) antes de colocar o objeto em um banco de dados, precisará chamar stripslashes( ) antes de desfazer a serialização da string.

Outro ponto a notar ao serializar classes ou utilizá-las como variáveis de sessão: o PHP precisa conhecer a estrutura de uma classe antes de poder instanciar a classe novamente. Portanto, você precisará incluir o arquivo de definição de classe antes de chamar session_start( ) ou unserialize( ).

Obtendo informações sobre o ambiente do PHP

Várias funções podem ser utilizadas para descobrir informações sobre como o PHP é configurado.

Descobrindo quais extensões estão carregadas

Você pode facilmente ver quais conjuntos de funções e quais funções estão disponíveis em cada um desses conjuntos utilizando as funções get_loaded_extensions( ) e get_extension_funcs( ).

A função get_loaded_extensions( ) retorna um array de todos os conjuntos de função atualmente disponíveis para o PHP. Dado o nome de um conjunto particular de funções ou extensão, get_extension_funcs( ) retorna um array das funções nesse conjunto.

O script na Listagem 23.1 lista todas as funções disponíveis para a instalação do PHP utilizando essas duas funções.

Listagem 23.1 **list_functions.php – Esse script lista todas as extensões disponíveis para o PHP e junto com cada extensão, fornece uma lista das funções nessa extensão**

```
<?php
  echo 'Function sets supported in this install are:<br />';
  $extensions = get_loaded_extensions( );
  foreach ($extensions as $each_ext)
  {
    echo "$each_ext <br />";
    echo '<ul>';
    $ext_funcs = get_extension_funcs($each_ext);
    foreach($ext_funcs as $func)
    {
       echo "<li> $func </li>";
    }
    echo '</ul>';
  }
?>
```

Observe que a função get_loaded_extensions( ) não aceita nenhum parâmetro, e a função get_extension_funcs( ) recebe o nome da extensão como seu único parâmetro.

Essas informações podem ser úteis se você estiver tentando dizer que instalou com sucesso uma extensão ou se estiver tentando gravar código portável que gere mensagens de diagnóstico úteis na instalação.

Identificando o proprietário do script

Você pode descobrir o usuário que possui o script em execução com uma chamada à função `get_current_user( )`, da seguinte maneira:

```
echo get_current_user( );
```

Isso às vezes pode ser útil para resolver questões de permissões.

Descobrindo quando o script foi modificado

Adicionar uma data da última modificação a cada página em um site é relativamente popular.

Você pode verificar a data da última modificação de um script com a função `getlastmod( )` (note a falta de sublinhados no nome da função), da seguinte maneira:

```
echo date('g:i a, j M Y',getlastmod( ));
```

A função retorna um registro de data/hora do Unix, que podemos alimentar com `date( )` como fizemos aqui, para produzir uma data legível por humanos.

Carregando extensões dinamicamente

Você pode realmente carregar bibliotecas de extensão em tempo de execução, se elas não estiverem compiladas, utilizando a função `dl( )`. Essa função espera como um parâmetro o nome do arquivo contendo a biblioteca. Sob o Unix, esses serão nomes de arquivo com a extensão de nome de arquivo `.so`; sob Windows, eles têm a extensão `.dll`.

Um exemplo de uma chamada a `dl( )` é:

```
dl('php_ftp.dll');
```

Isso dinamicamente carregará a extensão gd2 (geração de imagem) em uma máquina Windows.

Você não deve especificar o diretório em que o arquivo reside. Em vez disso, deve configurar no arquivo `php.ini`. Uma diretiva chamada `extension_dir` especificará o diretório onde PHP procurará bibliotecas para carregar dinamicamente.

Se achar que está tendo problemas em carregar extensões dinamicamente, verifique também a diretiva `enable_dl` em seu arquivo `php.ini`. Se ela estiver desativada, você não será capaz de carregar extensões de modo dinâmico. Particularmente, se a máquina em que você trabalha não for sua própria máquina, talvez isso esteja desativado por razões de segurança. Você também não será capaz de utilizar o `dl( )` se o PHP estiver executando no modo de segurança.

Alterando temporariamente o ambiente de tempo de execução

Você pode visualizar as diretivas configuradas no arquivo `php.ini` ou alterá-las durante o período de tempo de vida de um único script. Isso pode ser particularmente útil, por exemplo, em conjunção com a diretiva `max_execution_time` se você souber que seu script levará algum tempo para executar.

Você pode acessar e alterar as diretivas utilizando as funções gêmeas `ini_get( )` e `ini_set( )`. A Listagem 23.2 mostra um script simples que utiliza essas funções.

Listagem 23.2 **iniset.php – Esse script redefine as variáveis do arquivo php.ini**

```
<?php
  $old_max_execution_time = ini_set('max_execution_time', 120);
  echo "old timeout is $old_max_execution_time <br />";

  $max_execution_time = ini_get('max_execution_time');
  echo "new timeout is $max_execution_time <br />";
?>
```

A função ini_set( ) recebe dois parâmetros. O primeiro é o nome da diretiva de configuração contida no arquivo php.ini que desejamos alterar, e o segundo é o valor para o qual queremos alterá-lo. Ele retorna o valor anterior da diretiva.

Nesse caso, estamos redefinindo o valor do tempo máximo dos 30 segundos padrão para um script executar em 120 segundos.

A função ini_get( ) simplesmente verifica o valor de uma diretiva de configuração particular. O nome da diretiva deve ser passado para ela como uma string. Aqui, nós o estamos utilizando apenas para verificar se o valor foi realmente alterado.

Nem todas as opções INI podem ser definidas dessa maneira. Cada opção possui um nível em que pode ser configurada. Os possíveis níveis são:

- PHP_INI_USER – Você pode mudar esses valores em seus scripts com ini_set( ).
- PHP_INI_PERDIR – Mude esses valores em php.ini ou nos arquivos .htaccess ou httpd.conf se estiver utilizando o Apache. O fato de você poder modificá-los nos arquivos .htaccess significa que você pode mudar os valores por diretório – e, portanto, o nome.
- PHP_INI_SYSTEM – Mude esses valores nos arquivos php.ini ou httpd.conf.
- PHP_INI_ALL – É possível mudar esses valores de qualquer uma das seguintes maneiras – isto é, em um script, em um arquivo .htaccess ou em seus arquivos httpd.conf ou php.ini.

O conjunto completo de opções ini e os níveis nos quais podem ser definidas estão no manual do PHP, em http://www.php.net/ini_set.

Destacando a origem

O PHP vem com um recurso integrado de realce da sintaxe, semelhante a muitos ambientes de desenvolvimento integrado (Integrated Development Environments – IDEs). Em particular, ele é útil para compartilhar código com outros ou apresentá-lo para discussão em uma página Web.

As funções show_source( ) e highlight_file( ) são as mesmas. (A função show_source( ) é na realidade um alias para highlight_file( ).)Essas duas funções aceitam um nome de arquivo como parâmetro. (Esse arquivo deve ser um arquivo do PHP; caso contrário, você não obterá um resultado muito significativo.) Por exemplo:

```
show_source('list_functions.php');
```

O arquivo será ecoado para o navegador com o texto destacado em várias cores dependendo se ele é uma string, um comentário, uma palavra-chave ou HTML. A saída é impressa em uma cor de fundo. O conteúdo que não se encaixar em nenhuma dessas categorias é impresso em uma cor padrão.

A função highlight_string( ) opera de maneira semelhante, mas aceita uma string como parâmetro e a imprime no navegador em um formato de sintaxe destacada.

Você pode configurar as cores para destaque de sintaxe em seu arquivo php.ini. A seção que você está procurando parece o seguinte:

```
; Colors for Syntax Highlighting mode
highlight.string    =    #DD0000
highlight.comment   =    #FF8000
highlight.keyword   =    #007700
highlight.bg        =    #FFFFFF
highlight.default   =    #0000BB
highlight.html      =    #000000
```

As cores estão no formato padrão HTML RGB.

Utilizando o PHP na linha de comando

Você pode escrever ou fazer download de muitos programas pequenos e executá-los na linha de comando. Se você estiver em um sistema Unix, esses programas geralmente são escritos em uma linguagem de scripts shell ou Perl. Se você utiliza o sistema Windows, geralmente são escritos como arquivo de lote.

Provavelmente, você chegou ao PHP para um projeto Web, mas as mesmas facilidades de processamento de texto que o tornam uma poderosa linguagem para Web, também o tornam um excelente programa utilitário de linha de comando.

Existem três maneiras de executar um script de PHP na linha de comando: a partir de um arquivo, por meio de um pipe ou diretamente na linha de comando.

Para executar um script do PHP em um arquivo, certifique-se de que o executável do PHP (php ou php.exe, dependendo do seu sistema operacional) esteja no caminho e chame-o com o nome do script como argumento. Aqui está um exemplo:

```
php myscript.php
```

O arquivo myscript.php é apenas um arquivo PHP normal, portanto, contém qualquer sintaxe normal do PHP dentro das tags do PHP.

Para passar código por meio de um pipe, você pode executar qualquer programa que gere um script PHP válido como saída e passar ao executável php. O exemplo a seguir utiliza o programa echo para fornecer um programa de uma linha:

```
echo '<?php for($i=1; $i<10; $i++) echo $i; ?>' | php
```

Novamente, o código PHP está entre as tags do PHP (<?php e ?>). Observe também que esse é um programa de linha de comando echo, não a construção da linguagem do PHP.

Um programa de uma linha dessa natureza poderia ser mais fácil de passar diretamente a partir da linha de comando, como neste exemplo:

```
php -r 'for($i=1; $i<10; $i++) echo $i;'
```

A situação é um pouco diferente aqui. O código PHP passado nessa string não está entre as tags do PHP. Se você colocar a string entre as tags do PHP, receberá um erro de sintaxe.

São inúmeros os programas úteis de PHP que você pode escrever para uso na linha de comando. É possível escrever instaladores para aplicações PHP.Você pode criar um script rápido para reformatar um arquivo de texto antes de importá-lo ao banco de dados. E pode até criar um script para fazer tarefas repetitivas e necessárias na linha de comando; um bom candidato seria um script para copiar todos os arquivos PHP, imagens e estruturas de tabela MySQL a partir do servidor Web de teste para o de produção.

A seguir

A Parte V abrange vários projetos práticos relativamente complicados para utilizar o PHP e o MySQL. Esses projetos fornecem exemplos úteis para tarefas semelhantes que talvez você tenha de realizar e demonstram a utilização do PHP e do MySQL em projetos maiores.

O Capítulo 24 aborda algumas questões com que você se depara ao codificar projetos maiores utilizando o PHP. Essas questões incluem princípios de engenharia de software como projeto, documentação e gerenciamento de mudanças.

V

Construindo projetos práticos de PHP e MySQL

24

Utilizando o PHP e o MySQL para grandes projetos

NAS PRIMEIRAS PARTES DESTE LIVRO, discutimos vários componentes do PHP e do MySQL e utilizações para o PHP e MySQL. Embora tenhamos tentado tornar todos os exemplos interessantes e relevantes, eles tornaram-se muitos simples, consistindo em um ou dois scripts de até aproximadamente 100 linhas de código.

Quando você estiver construindo aplicações Web reais, raramente será assim tão simples. Há poucos anos, um Web site "interativo" tinha apenas um formulário de correio. Mas hoje em dia, os Web sites tornaram-se aplicações Web – isto é, um programa normal mas distribuído pela Web. Essa mudança de foco significa uma mudança em escala. Os Web sites crescem de uma porção de scripts para milhares e milhares de linhas de código. Os projetos desse tamanho requerem planejamento e gerenciamento como qualquer outro desenvolvimento de software.

Antes de mudarmos para a análise dos projetos desta seção do livro, examinaremos algumas técnicas que você pode utilizar para gerenciar projetos Web relativamente grandes. Essa é uma arte emergente e é obviamente difícil compreendê-la por inteiro: você pode ver isso observando o mercado.

Os tópicos-chave deste capítulo incluem:

- Como aplicar engenharia de software ao desenvolvimento na Web;
- Planejando e executando um projeto de aplicação na Web;
- Reutilizando o código;
- Escrevendo código sustentável;
- Implementando controle de versão;
- Escolhendo um ambiente de desenvolvimento;
- Documentando seu projeto;
- Prototipagem;
- Separando lógica, conteúdo e apresentação: PHP, HTML e CSS;
- Otimizando código.

Aplicando engenharia de software ao desenvolvimento para a Web

Como você provavelmente já sabe, a engenharia de software é a aplicação de uma abordagem quantificável e sistemática ao desenvolvimento de software. Isto é, a engenharia de software é a aplicação de princípios de engenharia ao desenvolvimento de software.

Além disso, a engenharia de software é infelizmente algo de que carecem muitos projetos Web. Isso por duas razões principais. A primeira razão é que o desenvolvimento na Web é freqüentemente gerenciado da mesma maneira que o desenvolvimento de relatórios escritos. É um exercício em estrutura de documento, projeto gráfico e produção. Esse é um paradigma orientado para documento. Isso é inteiramente satisfatório e bom para sites estáticos de pequeno e médio porte, mas à medida que aumentamos a quantidade de conteúdo dinâmico nos Web sites até o nível em que oferecem serviços em vez de documentos, esse paradigma não é mais apropriado. Muitas pessoas simplesmente não pensam em utilizar práticas de engenharia de software para um projeto Web.

A segunda razão pela qual as práticas de engenharia de software não são utilizadas é que o desenvolvimento de aplicações Web é diferente do desenvolvimento de aplicações normais sob vários aspectos. Lidamos com projetos que têm prazos de planejamento muito curtos, e sofremos uma pressão constante para ter o site construído *agora*. A engenharia de software está inteiramente relacionada a planejar e investir tempo no planejamento. Com projetos Web, é freqüente a percepção de que não temos o tempo de planejamento.

Quando não conseguimos planejar projetos Web, acabamos com os mesmos problemas como se não conseguíssemos planejar nenhum projeto de software: aplicações com falhas, não-cumprimento de prazos de entrega e código ilegível.

O truque, então, está em encontrar as partes da engenharia de software que funcionam nessa nova disciplina de desenvolvimento de aplicação Web e descartar as partes que não funcionam.

Planejando e executando um projeto de aplicação Web

Não há melhor metodologia ou ciclo de vida de projeto para projetos Web. Mas há várias considerações que você deve tomar ao fazer seu projeto. Listaremos essas aqui e discutiremos algumas delas em mais detalhe nas seções a seguir. Essas considerações estão em uma ordem específica, mas você não tem de segui-la se não se não se ajustar ao seu projeto. A ênfase aqui é estar ciente das questões e escolher técnicas que funcionarão para você.

- Antes de começar, pense no que você está tentando construir. Pense no objetivo final. Pense em quem vai utilizar sua aplicação Web; isto é, sua audiência-alvo. Muitos projetos Web que são tecnicamente perfeitos falham porque ninguém verificou se havia usuários interessados em tal aplicação.

- Tente dividir sua aplicação em componentes. Que partes ou passos de processo sua aplicação tem? Como cada um desses componentes funcionará? Como eles se ajustarão? Desenhar cenários, storyboards ou mesmo utilizar casos pode ser útil para você ter uma idéia disso.

- Depois que tiver uma lista de componentes, veja qual desses já existem. Se um módulo pré-escrito tiver essa funcionalidade, procure utilizá-lo. Não se esqueça de procurar o código existente dentro e fora da sua organização. Em particular na comunidade Open Source (de código-fonte aberto), muitos componentes preexistentes de código estão livremente disponíveis para utilização. Decida que código você precisa escrever a partir do zero e qual será aproximadamente o tamanho desse trabalho.

- Tome decisões sobre questões de processo. Isso é freqüentemente ignorado em projetos Web. Questões de processo significam padrões de codificação, estruturas de diretórios, gerenciamento de controle de versão, ambiente de desenvolvimento, nível e padrões de documentação e alocações de tarefa para membros de equipe.

- Construa um protótipo, baseado em todas as informações anteriores. Mostre-o para usuários. Itere.
- Lembre-se de que, nisso tudo, é importante e útil separar conteúdo e lógica da aplicação. Explicaremos essa idéia em mais detalhe em um minuto.
- Faça todas as otimizações que achar necessárias.
- À medida que avançar, teste, com tanto cuidado quanto testaria qualquer projeto de desenvolvimento de software.

Reutilizando código

Os programadores freqüentemente cometem o erro de reescrever código que já existe. Quando você sabe quais componentes de aplicação são necessários ou, em uma escala menor, quais funções são necessárias, verifique o que está disponível antes de iniciar o desenvolvimento.

Um dos poderes do PHP como uma linguagem é sua grande biblioteca de funções predefinidas. Sempre verifique se existe uma função que faz aquilo que você está tentando fazer. Normalmente, não é muito difícil encontrar o que você quer. Uma boa maneira de fazer isso é pesquisar o manual por grupo de função.

Às vezes os programadores reescrevem funções acidentalmente porque não examinaram o manual para ver se uma função existente fornece a funcionalidade de que eles precisam. Sempre mantenha um bookmark para o manual se você estiver on-line ou faça download da versão atual e navegue localmente. Entretanto, note que o manual on-line é freqüentemente atualizado e há a vantagem de ser capaz de navegar no manual editado. O manual editado é um recurso fantástico uma vez que contém comentários, sugestões e código de exemplo de outros usuários que freqüentemente respondem às mesmas perguntas que você pode ter depois de ler a página básica do manual. O manual costuma ter relatórios de bugs e soluções antes que eles sejam consertados e documentados no corpo da documentação.

Você pode conseguir a versão no idioma inglês em:

http://www.php.net/manual/en/

Alguns programadores que falam um idioma diferente do inglês talvez sejam tentados a escrever funções empacotadoras para essencialmente renomear funções do PHP para corresponder ao idioma que conhecem. Essa prática às vezes é chamada de "açúcar sintático". Essa é uma má idéia – ela tornará a leitura e a manutenção do seu código mais difícil para outros. Se estiver aprendendo uma nova linguagem, você deve aprender como utilizá-la adequadamente. Além disso, adicionar um nível de chamada de função dessa maneira tornará o código lento. Considerando tudo, essa é uma abordagem que deve ser evitada.

Se achar que a funcionalidade de que você precisa não está na biblioteca principal do PHP, você tem duas escolhas. Se precisar de algo muito simples, você pode escolher escrever sua própria função ou objeto. Entretanto, se estiver examinando uma parte relativamente complexa de funcionalidade na construção – como um carrinho de compras, um sistema de e-mail Web ou fóruns Web – você não se surpreenderá ao ver que esses provavelmente já foram construídos por uma outra pessoa. Uma das forças de trabalhar na comunidade Open Source é que o código para componentes de aplicações como essas com freqüência está livremente disponível. Se encontrar um componente semelhante à aplicação que deseja construir, mesmo se não for exatamente adequado, você pode ver o código-fonte como um ponto inicial para modificação ou para construir o seu próprio código.

Se acabar desenvolvendo suas próprias funções ou componentes, você deve seriamente considerar disponibilizá-los para a comunidade do PHP quando concluir. Esse é o princípio que mantém a comunidade de desenvolvedores do PHP um grupo tão informado, ativo e útil.

Escrevendo código sustentável

A questão da capacidade de manutenção é freqüentemente negligenciada em aplicações Web, particularmente porque quase sempre escrevemos com pressa. A introdução ao código e sua conclusão às vezes parece mais importante que planejá-lo primeiro. Mas investir um pouco de tempo pode poupar bastante tempo mais à frente quando você estiver para construir a próxima iteração de uma aplicação.

Padrões de codificação

A maioria das empresas de IT tem padrões de codificação – diretrizes para o estilo de escolha de nomes de arquivos e variáveis, diretrizes para comentar código, diretrizes para recuar código e assim por diante.

Devido ao paradigma do documento antigamente aplicado com freqüência ao desenvolvimento Web, os padrões de codificação às vezes eram negligenciados nessa área. Se estiver codificando por conta própria ou em uma pequena equipe, é fácil subestimar a importância de padrões de codificação. Não faça isso porque sua equipe e seu projeto podem crescer. Portanto, você não apenas acabará se confundindo, como também confundirá vários programadores que podem não entender nada do código existente.

Definindo convenção para atribuição de nomes

Os objetivos de definir uma convenção para atribuição de nomes são:

- Tornar o código fácil para a leitura. Se definir nomes de variáveis e funções sensatamente, você deve conseguir ler código praticamente da maneira como leria uma frase em seu idioma, ou pelo menos pseudocódigo.
- Tornar nomes de identificadores fáceis de lembrar. Se os identificadores forem formatados consistentemente, será mais fácil lembrar do que você chamou uma variável ou função particular.

Os nomes de variável devem descrever os dados que eles contêm. Se estiver armazenando o sobrenome de alguém, chame-o de `$sobrenome`. Você precisa achar um equilíbrio entre comprimento e legibilidade. Por exemplo, armazenar o nome em `$s` facilitará a digitação, mas será difícil entender o código. Armazenar o nome em `$sobrenome_do_usuario_atual` é mais informativo, mas é preciso digitar mais (portanto, é mais fácil cometer um erro de digitação) e realmente não acrescenta tanto.

Você precisa tomar uma decisão sobre a utilização de letras maiúsculas e minúsculas. Os nomes de variável diferenciam letras maiúsculas de minúsculas no PHP, como mencionamos antes. Você precisa decidir se seus nomes de variável serão inteiramente em minúsculas, em maiúsculas ou uma combinação delas, por exemplo, colocar em letras maiúsculas as primeiras letras das palavras. Somos propensos a utilizar tudo em letras minúsculas, já que é algo fácil de lembrar.

Também é uma boa idéia distinguir entre variáveis e constantes com maiúsculas/minúsculas – um esquema comum é utilizar tudo em letras minúsculas para variáveis (por exemplo, `$result`) e tudo em letras maiúsculas para constantes (por exemplo, `PI`).

Uma prática ruim que alguns programadores utilizam é ter duas variáveis com o mesmo nome mas com o uso de letras maiúsculas e minúsculas diferentes, só porque eles podem, como `$nome` e `$Nome` por exemplo. Espero que seja óbvio por que essa é uma idéia terrível.

Além disso, é melhor evitar esquemas humorísticos de uso de letras maiúsculas e minúsculas como `$WaReZ` porque ninguém será capaz de lembrar como isso funciona.

Você também deve pensar sobre o esquema a utilizar para nomes de variáveis com diversas palavras. Por exemplo, já vi todos estes esquemas utilizados:

```
$username
$user_name
$UserName
```

Não importa por qual você opte, deve tentar ser consistente em relação a isso. Talvez você também queira impor um limite máximo sensato de duas a três palavras em um nome de variável.

Os nomes de função têm muitas das mesmas considerações, com algumas extras. Os nomes de função geralmente devem ser baseados no verbo. Considere funções predefinidas do PHP como `addslashes( )` ou `mysql_connect( )`, que descrevem o que elas vão fazer para ou com os parâmetros que lhes são passados. Isso aprimora de modo excelente a legibilidade do código. Note que essas duas funções têm um esquema de nomeação diferente para lidar com nomes de função de várias palavras. Quanto a isso, as funções do PHP são inconsistentes, presumivelmente, em parte, devido ao fato de serem escritas por um grande número de pessoas, mas principalmente porque muitos nomes de funções foram adotados sem alterações a partir de vários idiomas e APIs diferentes.

Lembre-se também de que nomes de função não fazem distinção entre letras maiúsculas e minúsculas no PHP. De qualquer maneira, provavelmente você deve ater-se a um formato particular, apenas para evitar confusão.

Você também poderia querer considerar a utilização do esquema de atribuição de módulo utilizado em muitos módulos do PHP, isto é, prefixar o nome de funções com o nome do módulo. Por exemplo, todas as funções do MySQL iniciam com `mysql_` e todas as funções IMAP iniciam com `imap_`. Se, por exemplo, você tiver um módulo de carrinho de compras (shopping cart) no código, poderia prefixar a função nesse módulo com `cart_`.

Observe, entretanto, que quando o PHP5 fornece tanto uma interface procedural como uma orientada a objeto, os nomes da função são diferentes. Geralmente, as procedurais usam sublinhados (`my_function( )`) e as orientadas a objeto usam o que chamamos studlyCaps (`myFunction( )`).

No final, realmente não importa que convenções e padrões você utiliza ao escrever o código, contanto que algumas diretrizes consistentes sejam aplicadas.

Comentando seu código

Todos os programas devem ser comentados até um nível sensato. Talvez você pergunte qual nível de comentário poderia ser considerado sensato. Geralmente, você deve considerar adicionar um comentário a cada um dos seguintes itens:

- **Arquivos, quer sejam scripts completos ou arquivos include.** Cada arquivo deve ter um comentário declarando que arquivo é esse, para que serve, quem o escreveu e quando foi atualizado.

- **Funções.** Os comentários de função devem especificar o que faz a função, que entrada ela espera e o que ela retorna.

- **Classes.** Os comentários devem descrever o propósito da classe. Os métodos de classe devem ter o mesmo tipo e nível de comentários que qualquer outra função.

- **Fragmentos de código dentro de um script ou função.** Freqüentemente acho útil escrever um script começando com um conjunto de comentários no estilo de pseudocódigo e depois preencher o código para cada seção. Então, um script inicial talvez seja semelhante a:

  ```
  <?
  // valida os dados de entrada
  // envia para o banco de dados
  // reporta os resultados
  ?>
  ```

  Isso é bem útil porque depois que você preencheu todas as seções com chamadas de função ou com o que for, seu código já está comentado.

- **Código complexo ou hacks.** Quando demora o dia todo para você fazer algo ou se tiver de fazer isso de uma maneira esquisita, escreva um comentário que explique por que você fez isso dessa maneira. Assim, da próxima vez que vir o código, você não precisará bater a cabeça pensando: "Ai, Meu Deus, para que serve isso?"

Outra diretriz geral a seguir é comentar à medida que você avança. Você poderia pensar que voltará e comentará o código quando tiver concluído um projeto. Garanto que isso não acontecerá, a menos que você tenha bem menos horários de desenvolvimento punitivos e mais autodisciplina que nós.

Recuando

Como em qualquer linguagem de programação, você deve recuar o código de modo consistente e sensato. É como planejar um currículo ou uma carta comercial. O recuo torna o código mais fácil de ler e mais rápido de entender.

Em geral, qualquer bloco de programa que pertença internamente a uma estrutura de controle deve ser recuado a partir do código adjacente. O grau de recuo deve ser notável (isto é, mais de um espaço) mas não excessivo. Geralmente, penso que o uso de tabulações deve ser evitado. Embora fácil de digitar, elas consomem bastante espaço de tela nos monitores de muitas pessoas. Utilizamos um nível de recuo de dois a três espaços para todos os projetos.

A maneira como você dispõe suas chaves também é uma questão. Os dois esquemas mais comuns são:

Esquema 1:

```
if (condição) {
  // faz algo
}
```

Esquema 2:

```
if (condição)
{
  // faz algo mais
}
```

Qual deles utilizar depende de você. O esquema que você escolhe deve (novamente) ser utilizado de forma consistente por todo um projeto para evitar confusão.

Dividindo código

Um código gigante, monolítico é horrível. Algumas pessoas terão um script enorme que faz tudo em uma linha de código principal. É indiscutivelmente melhor dividir o código em funções e/ou classes e posicionar itens relacionados dentro de arquivos include. Você pode, por exemplo, colocar todas as suas funções relacionadas ao banco de dados em um arquivo chamado `dbfunctions.php`.

As razões para dividir o código em partes racionais incluem:

- Tornar o código mais fácil de ler e entender.
- Tornar o código mais reutilizável e minimizar redundância. Por exemplo, você poderia reutilizar o arquivo `dbfunctions.php` anterior em cada script em que precisar estabelecer uma conexão com seu banco de dados. Se precisar alterar a maneira como isso funciona, você tem de alterá-la só em um lugar.
- Facilitar o trabalho em equipe. Se o código for dividido em componentes, você poderá atribuir responsabilidade sobre os componentes a membros de equipe. Isso também significa que você pode evitar a situação em que um programador está esperando outro terminar de trabalhar no `GiantScript.php` para que ele possa continuar com seu próprio trabalho.

No início de um projeto, você deve gastar algum tempo pensando sobre como vai dividir um projeto em componentes planejados. Isso requer delimitar as áreas de funcionalidade, mas não perca muito tempo com isso uma vez que o esquema pode mudar depois que você colocar um projeto em funcionamento. Você também precisará decidir quais componentes precisam ser construídos primeiro, quais componentes dependem de outros componentes e a linha de tempo para desenvolver todos eles.

Mesmo que todos os membros da equipe forem trabalhar em todas as partes do código, geralmente é uma boa idéia atribuir a responsabilidade principal sobre cada componente a uma pessoa específica. Em última instância, essa seria a pessoa responsável se algo sair errado com o componente. Alguém também deve assumir o trabalho de gerente da obra – isto é, a pessoa que se certifica de que todos os componentes estão no caminho certo e funcionando com os demais componentes. Essa pessoa normalmente também gerencia controle de versão – conversaremos sobre isso mais tarde no capítulo. Essa pessoa pode ser o gerente de projeto ou designada para uma responsabilidade separada.

Utilizando uma estrutura de diretórios padrão

Ao iniciar um projeto, você precisa pensar sobre como a estrutura de componente será refletida na estrutura de diretórios do Web site. Assim como é uma má idéia ter um script gigante contendo toda a funcionalidade, normalmente é uma má idéia ter um diretório gigante contendo tudo. Decida como você vai dividi-lo entre componentes, lógica, conteúdo e bibliotecas compartilhadas de código. Documente a estrutura e certifique-se de que todos que trabalham no projeto tenham uma cópia para que possam encontrar o que procuram. Isso leva ao próximo ponto.

Documentando e compartilhando funções internamente

À medida que você desenvolve bibliotecas de funções, torne-as disponíveis para outros programadores na sua equipe. Comumente, cada programador em uma equipe escreve seu próprio conjunto de funções de banco de dados, data ou depuração. Isso é um desperdício de tempo. Disponibilize as funções e classes para outros.

Lembre-se de que mesmo que o código seja armazenado em uma área ou diretório comumente disponível para a sua equipe, eles não saberão que isso existe a menos que sejam informados. Desenvolva um sistema para documentar internamente as bibliotecas de funções, e disponibilize-o para programadores na sua equipe.

Implementando o controle da versão

Controle de versão é a arte de gerenciamento de alterações concorrentes aplicada ao desenvolvimento de software. Os sistemas de controle de versão geralmente atuam como um *repositório* ou arquivo central e fornecem uma interface controlada para acessar e compartilhar o código (e possivelmente a documentação).

Imagine uma situação em que você esteja tentando melhorar um código, mas em vez disso o estrague acidentalmente e não consiga restaurá-lo, independente do seu esforço. Ou, você ou um cliente decide que uma versão anterior do site era melhor. Ou, você precisa restaurar uma versão anterior por razões jurídicas.

Imagine outra situação em que dois membros de sua equipe de programação querem trabalhar no mesmo arquivo. Eles dois poderiam abrir e editar o arquivo ao mesmo tempo, sobrescrevendo as alterações um do outro. Ambos poderiam ter uma cópia local que poderiam desenvolver e alterar de diferentes maneiras. Se você considerar essas possibilidades, um programador pode ficar sem fazer nada enquanto espera outro programador terminar de editar um arquivo.

Você pode resolver todos esses problemas com um sistema de controle de versão. Esses sistemas podem monitorar alterações em cada arquivo no repositório de modo que você possa ver não só o

estado atual de um arquivo, como também a aparência que ele tinha em uma determinada data/hora no passado. Esse recurso permite reverter o código danificado para uma versão reconhecidamente funcional. Você pode rotular um conjunto particular de instâncias de um arquivo como uma versão de lançamento, querendo dizer que você pode continuar o desenvolvimento no código, e obter acesso a uma cópia da versão atualmente lançada a qualquer momento.

Os sistemas de controle de versão também ajudam vários programadores a trabalharem juntos no código. Cada programador pode obter uma cópia do código no repositório (o que é chamado de *fazer o check-out*) e quando eles fizerem as alterações, essas alterações podem ser mescladas de volta no repositório (que é chamada de *fazer o check-in* ou "*comitar*", do termo inglês *commit*). Os sistemas de controle de versão, portanto, podem monitorar quem fez cada alteração em um sistema.

Esses sistemas normalmente têm um recurso para gerenciar atualizações concorrentes. O que isso significa é que dois programadores realmente podem modificar o mesmo arquivo ao mesmo tempo. Por exemplo, imagine que tanto John como Mary tenham feito o check-out de uma cópia da versão mais recente do projeto. John conclui suas alterações em um arquivo particular e faz o check-in. Mary também altera esse arquivo e também tenta fazer o check-in. Se as alterações que eles fizeram não estiverem na mesma parte do arquivo, o sistema de controle de versão mesclará as duas versões do arquivo. Se as alterações entrarem em conflito, Mary será notificada e as duas versões diferentes serão apresentadas. Ela então pode ajustar sua versão do código para evitar os conflitos.

O sistema de controle de versão utilizado pela maioria dos desenvolvedores Unix e/ou Open Source é o CVS, que significa Concurrent Versions System. O CVS tem código-fonte aberto, vem de bonificação com praticamente todo Unix e você também pode obtê-lo para PCs, executando DOS ou Windows, e Mac. Ele suporta um modelo cliente-servidor que permite fazer o check-in ou check-out do código a partir de qualquer máquina com uma conexão Internet, assumindo que o servidor do CVS seja visível na rede. Ele é utilizado para o desenvolvimento do PHP, Mozilla, Apache, entre outros projetos de alto nível, pelo menos parcialmente por essa razão.

Você pode fazer download do CVS para seu sistema na home page do CVS em:

http://www.cvshome.org/

Embora o sistema CVS básico seja uma ferramenta de linha de comando, vários suplementos oferecem um front-end mais elegante, incluindo front-ends baseados em Java e Windows. Esses também podem ser acessados na home page do CVS.

Bitkeeper é um produto rival de controle de versão, usado por alguns projetos de código aberto de alto nível, incluindo MySQL e o kernel do Linux. Está disponível gratuitamente para projetos de código aberto.

Há alternativas comerciais ao CVS. Um desses é o perforce, que é executado na maioria das plataformas comuns e tem suporte ao PHP. Observe que embora o programa seja comercial, licenças gratuitas são oferecidas para projetos com código-fonte aberto em:

http://www.perforce.com/

Escolhendo um ambiente de desenvolvimento

A discussão sobre controle de versão nos leva ao tópico mais geral dos ambientes de desenvolvimento. Tudo o que você realmente necessita é de um editor de textos e um navegador para teste, mas freqüentemente os programadores são mais produtivos em um ambiente integrado (IDE).

Há um número crescente de projetos gratuitos para construir um PHP IDE dedicado, incluindo o KPHPDevelop, para o ambiente da área de trabalho KDE no Linux, disponível em:

http://kphpdev.sourceforge.net/

Atualmente, todos os melhores PHP IDEs são comerciais. O Zend Studio da zend.com, o Komodo da activestate.com e o PHPEd da nusphere.com fornecem IDEs com muitos recursos. Todos têm uma versão de avaliação para download, mas exigem pagamento para utilização posterior. O Komodo tem uma licença de uso não-comercial muito barata.

Documentando seus projetos

Você pode produzir muitos tipos diferentes de documentação para seus projetos de programação, incluindo, mas não limitado, aos seguintes:

- Documentação de projeto;
- Documentação técnica/guia do desenvolvedor;
- Dicionário de dados (incluindo documentação de classe);
- Guia do usuário (embora a maioria das aplicações Web tenha de ser autoexplicativa).

Nosso objetivo aqui não é ensinar a escrever a documentação técnica, mas sugerir que você torne sua vida mais fácil automatizando parte do processo.

Em algumas linguagens, há maneiras de gerar automaticamente alguns desses documentos – particularmente a documentação técnica e dicionários de dados. Por exemplo, o javadoc gera uma árvore de arquivos HTML contendo protótipos e descrições de membros de classe para programas Java.

Um grande número de utilitários desse tipo está disponível para o PHP. Algum desses são:

- phpdoc, disponível em:

  http://www.phpdoc.de/

  Esse é o sistema utilizado pelo PEAR para documentar o código. Note que o termo *phpDoc* é utilizado para descrever vários projetos desse tipo, dos quais esse é um.

- PHPDocumentor, disponível em:

  http://phpdocu.sourceforge.net

  O PHPDocumentor gera uma saída muito semelhante ao javadoc e parece funcionar de maneira bastante resistente. Parece que ele tem uma equipe mais ativa de desenvolvedores do que os dois que listamos aqui.

- phpautodoc, disponível em:

  http://sourceforge.net/projects/phpautodoc/

  Novamente, o phpautodoc produz saída semelhante ao javadoc.

Um bom lugar para procurar mais aplicações desse tipo (e componentes do PHP em geral) é em SourceForge:

http://sourceforge.net

O SourceForge é principalmente utilizado pela comunidade de UNIX/Linux, mas também há muitos projetos para outras plataformas.

Prototipagem

A *prototipagem* é o ciclo de vida de um desenvolvimento comumente utilizado para desenvolver aplicações Web. Um protótipo é uma ferramenta útil para elaborar os requisitos do cliente. Normalmente é uma versão funcional parcialmente simplificada de uma aplicação que pode ser utilizada para uma discussão com o cliente e como a base do sistema final. Freqüentemente, as diversas iterações sobre um protótipo produzem a aplicação final. A vantagem dessa abordagem é que ela permite trabalhar de perto com um cliente ou usuário final para produzir um sistema com que eles estarão satisfeitos e do qual terão alguma propriedade.

Para ser capaz de produzir um protótipo rapidamente, você precisará de algumas ferramentas e habilidades particulares. Isso é onde uma abordagem baseada em componente funciona bem. Se tiver acesso a um conjunto de componentes preexistentes, disponível tanto pública como interna-

mente, você será capaz de fazer isso muito mais rapidamente. Outra ferramenta útil para rápido desenvolvimento de protótipos são os modelos. Veremos estes na próxima seção.

Há dois problemas principais em utilizar uma abordagem de prototipagem. Você precisa estar ciente de quais são esses problemas para que possa evitá-los e utilizar essa abordagem em seu potencial máximo.

O primeiro problema é que os programadores freqüentemente acham difícil jogar fora o código que escreveram por uma razão ou outra. Os protótipos são com freqüência escritos rapidamente e, do ponto de vista do projeto final, é possível ver que você não construiu um protótipo da melhor maneira possível, ou nem mesmo perto disso. Seções do código podem ser corrigidas, mas se a estrutura total estiver errada, você tem problemas. O problema é que aplicações Web são freqüentemente construídas sob enorme pressão de prazo e talvez não haja tempo de corrigi-las. Então você fica preso a um sistema mal projetado cuja manutenção é muito complicada.

Você pode evitar isso fazendo um pequeno planejamento como discutimos anteriormente neste capítulo. Lembre-se também de que às vezes é mais fácil desfazer-se de algo e começar tudo de novo do que corrigir. Embora isso talvez pareça algo para o que você não tem tempo, freqüentemente poupa muitos problemas mais tarde.

O segundo problema com a prototipagem é que um sistema pode acabar sendo um protótipo eterno. Toda vez que você pensar que terminou, seu cliente sugerirá mais algumas melhorias, funcionalidades adicionais ou atualizações para o site. Tudo isso pode fazer com que você nunca termine um projeto.

Para evitar esse problema, desenhe um plano de projeto com um número fixo de iterações e uma data após a qual nenhuma nova funcionalidade possa ser adicionada sem fazer um novo planejamento, um novo orçamento e um novo cronograma.

Separando lógica e conteúdo

Você provavelmente está familiarizado com a idéia de utilizar HTML para descrever a estrutura de um documento Web e utilizar *folhas de estilo em cascata (cascading style sheets – CSS)* para descrever sua aparência. Essa idéia de separar a apresentação do conteúdo pode ser estendida ao script. Em geral, os sites serão mais fáceis de utilizar e manter a longo prazo se você puder separar a lógica do conteúdo da apresentação. Isso se resume a separar o código em PHP do código em HTML.

Para projetos simples com um pequeno número de linhas de código ou scripts, isso pode mais atrapalhar do que valer a pena. Como seus projetos aumentam, é essencial encontrar um caminho de separar lógica e conteúdo. Se não fizer isso, seu código se tornará cada vez mais difícil de manter. Se você, seus superiores ou seu cliente decidirem aplicar um novo projeto ao Web site e uma grande quantidade de HTML for embutida no código, mudar o projeto será um pesadelo.

As três abordagens básicas para separar lógica e conteúdo são:

- Utilize arquivos include para armazenar diferentes partes do conteúdo. Essa é uma abordagem simplista, mas se seu site for principalmente estático, pode funcionar bem. Esse tipo de abordagem foi explicado no exemplo da TLA Consulting no Capítulo 5.
- Utilize uma função ou classe API com um conjunto de funções de membro para conectar conteúdo dinâmico em modelos estáticos de página. Vimos essa abordagem no Capítulo 6.
- Utilize um sistema de modelo. Esses analisam sintaticamente os modelos estáticos e utilizam expressões regulares para substituir tags marcadoras de lugar por dados dinâmicos. A principal vantagem disso é que se outro profissional projetar seus modelos, como um designer gráfico, ele não precisará conhecer absolutamente nada sobre o código do PHP. Você deve ser capaz de utilizar modelos fornecidos com modificação mínima.

Vários sistemas de modelo estão disponíveis. Provavelmente o mais popular seja o Smarty, disponível em:

http://smarty.php.net/

Otimizando código

Se você tiver experiência com programação não-Web, a otimização pode parecer realmente importante. Ao utilizar o PHP, a maior parte do tempo que um usuário espera uma aplicação Web está associada ao tempo de conexão e ao tempo de download. A otimização de seu código terá pouco efeito sobre esses tempos.

Utilizando otimizações simples

Há, porém, algumas otimizações simples que podem ser feitas que farão diferença. Muitas dessas se relacionam a aplicações que integram um banco de dados como o MySQL com seu código do PHP e algumas são listadas a seguir:

- Reduza conexões de banco de dados. Conectar-se a um banco de dados é freqüentemente a parte lenta de qualquer script. Você pode evitar isso utilizando conexões persistentes.
- Acelere as consultas de banco de dados. Reduza o número de consultas que você faz e certifique-se de que elas foram otimizadas. Com uma consulta complexa (e, portanto, lenta), normalmente há mais de uma maneira de fazer isso. Execute suas consultas a partir da interface de linha de comandos do banco de dados e experimente abordagens diferentes para acelerar o processo. No MySQL, você pode utilizar a instrução `EXPLAIN` para ver onde uma consulta poderia estar demorando. (A utilização dessa instrução é discutida no Capítulo 12.) Em geral, o princípio é minimizar junções e maximizar a utilização de índices.
- Minimize a geração de conteúdo estático a partir do PHP. Se todo fragmento de HTML que você produzir vier de `echo` ou `print( )`, isso exigirá muito mais tempo. (Esse é um dos argumentos para mudar a linha de pensamento e adotar uma separação entre lógica e conteúdo como descrito anteriormente.) Isso também se aplica a gerar dinamicamente imagens de botão – talvez você queira utilizar o PHP para gerar os botões uma vez e então reutilizá-los conforme necessário. Se você estiver gerando páginas puramente estáticas a partir de funções ou modelos toda vez que uma página carregar, considere executar as funções ou utilizar os modelos uma vez e salvar o resultado.
- Utilize as funções de string em vez de expressões regulares sempre que possível. Elas são mais rápidas.
- Utilize onde for possível strings de aspas simples em vez de strings de aspas duplas. O PHP avalia strings de aspas duplas, procurando variáveis para substituição. As strings de aspas simples não são avaliadas. Por outro lado, se ela estiver entre aspas simples, provavelmente o conteúdo é estático. Revise o que está fazendo e veja se pode eliminar completamente a string transformando-a em HTML estática.

Utilizando produtos Zend

A Zend Technologies é proprietária do mecanismo de script do PHP utilizado (código-fonte aberto) do PHP 4 em diante. Além do mecanismo básico, você também pode fazer download do Zend Optimizer. Esse é um otimizador de múltiplas passagens que otimizará seu código e poderá aumentar a velocidade em que seus scripts executam de 40% a 100%. Você precisará do PHP 4.0.2 ou superior para executar o otimizador. Embora de fonte fechada, seu download é gratuito no site da Zend:

http://www.zend.com

Esse suplemento funciona otimizando o código produzido pela compilação do tempo de execução do script. Outros produtos da Zend incluem: Zend Studio, Zend Accelerator, Zend Encoder e contratos comerciais de suporte.

Testando

Revisar e testar código é outro ponto básico da engenharia de software que é freqüentemente negligenciado no desenvolvimento Web. É muito fácil tentar executar o sistema com dois ou três casos de teste e então dizer, "legal, funciona bem". Esse é um equívoco comumente cometido. Certifique-se de que você testou extensamente e revisou vários cenários antes de considerar pronto o projeto.

Sugerimos duas abordagens que podem ser utilizadas para reduzir o nível de bugs do código. (Você nunca pode eliminar completamente os bugs; mas certamente pode eliminar ou minimizar a maioria deles.)

Primeiro, adote uma prática de revisão de código. Esse é o processo em que outro programador ou equipe de programadores examina seu código e sugere melhoras. Esse tipo de análise freqüentemente sugerirá:

- Erros que você não enxergou;
- Casos de teste sobre os quais você não pensou;
- Otimização;
- Melhoras em segurança;
- Componentes existentes que você poderia utilizar para melhorar uma parte de código;
- Funcionalidade adicional.

Mesmo que você trabalhe sozinho, pode ser bom encontrar um "companheiro de código" que esteja na mesma situação, e um revisar o código do outro.

A segunda sugestão é encontrar testadores para suas aplicações Web que representem os usuários finais do produto. A principal diferença entre aplicações Web e aplicações de área de trabalho é que todo mundo utilizará aplicações Web. Você não deve fazer suposições que usuários conhecerão computadores. Você não pode fornecer um grosso manual ou um cartão de referência rápida. Em vez disso, você tem de tornar as aplicações Web autodocumentadas e auto-explicativas. Você deve pensar nas maneiras como os usuários devem querer utilizar sua aplicação. A usabilidade é o principal parâmetro.

Pode ser realmente difícil entender os problemas que usuários finais ingênuos encontrarão se você for um experiente programador ou usuário Web. Uma maneira de abordar isso é obter testadores que representem o usuário típico.

Uma maneira como fizemos isso no passado foi liberar aplicações Web somente como uma beta. Quando você acreditar que eliminou a maioria dos bugs, publique a aplicação para um pequeno grupo de usuários de teste e obtenha um pequeno volume de tráfego pelo site. Ofereça serviços para os primeiros 100 usuários que retornarem um feedback sobre o site. Garanto que eles fornecerão alguma combinação de dados ou funcionalidades que você nunca imaginou. Se estiver construindo um Web site para a empresa de um cliente, a empresa pode freqüentemente fornecer um bom conjunto de usuários ingênuos a partir do quadro de funcionários da empresa para navegar pelo site. (Isso tem o benefício intrínseco de aumentar um sentido de posse do cliente sobre um site.)

Leitura adicional

Há muito material abrangendo essa área – estamos falando basicamente da ciência de engenharia de software, sobre a qual inúmeros livros foram escritos.

Um excelente livro que explica a dicotomia de um Web site como um documento *versus* um Web site como aplicação é *Web Site Engineering: Beyond Web Page Design,* de Thomas A. Powell. Qualquer livro de engenharia de software que você gostar servirá como um backup.

Para obter nformações sobre controle de versão, visite o Web site do CVS:
http://www.cvshome.org

Não há muitos livros sobre controle de versão (isso é surpreendente considerando o quanto isso é importante!), mas você pode tentar tanto o *Open Source Development with CVS,* de Karl Franz Fogel, como o *CVS Pocket Reference*, de Gregor N. Purdy.

Se estiver procurando componentes do PHP, IDEs ou sistemas de documentação, tente o SourceForge:

http://sourceforge.net

Muitos dos tópicos abordados neste capítulo são discutidos em artigos no site do Zend. Você poderia considerar uma visita a esse site para informações adicionais sobre o assunto. Você também poderia considerar fazer o download do otimizador do site quando estiver lá.

http://www.zend.com

Se você achou este capítulo interessante, talvez queira ver Extreme Programming, que é uma metodologia de desenvolvimento de software voltada para domínios onde os requisitos mudam freqüentemente, como desenvolvimento Web. O Web site da Extreme Programming está em:

http://www.extremeprogramming.org

A seguir

No Capítulo 25, veremos diferentes tipos de erros de programação, mensagens de erro do PHP e técnicas para localizar e tratar erros de maneira elegante.

25

Depurando

ESTE CAPÍTULO TRATARÁ DA DEPURAÇÃO de scripts de PHP. Se tiver examinado alguns exemplos no livro ou utilizado o PHP antes, provavelmente você já desenvolveu algumas técnicas e habilidades de depuração próprias. À medida que os projetos tornam-se mais complexos, a depuração pode tornar-se mais difícil. Embora suas habilidades melhorem, é mais provável que os erros envolvam diversos arquivos ou interações entre o código de várias pessoas.

Os tópicos-chave neste capítulo incluem:

- Tipos de erro de programação;
  - Erros de sintaxe;
  - Erros de tempo de execução;
  - Erros de lógica;
- Mensagens de erro;
- Níveis de erro;
- Desencadeando seus próprios erros;
- Tratando erros de maneira elegante.

Erros de programação

Independente da linguagem utilizada, há três tipos de erros gerais de programa:

- Erros de sintaxe;
- Erros de tempo de execução;
- Erros de lógica.

Examinaremos brevemente cada um antes de discutirmos algumas táticas para detectar, tratar, evitar e resolver erros.

Erro de sintaxe

As linguagens de programação, assim como os idiomas humanos, têm um conjunto de regras chamado *sintaxe* de uma linguagem/língua, que as instruções/orações devem seguir a fim de serem válidas. Isso se aplica tanto aos idiomas naturais, como o inglês, como às linguagens de programação, como o PHP. Se uma oração/instrução não seguir as regras de uma língua/linguagem, diz-se que ela contém um erro de sintaxe. Os erros de sintaxe também são freqüentemente chamados erros de

analisador de sintaxe quando se fala de linguagens interpretadas, como o PHP, ou erros de compilador quando se trata de linguagens compiladas, como C ou Java.

Se desobedecermos às regras de sintaxe da língua inglesa, ainda há uma boa chance de as pessoas saberem o que pretendemos dizer. Isso normalmente não é o caso com linguagens de programação. Se um script não seguir as regras de sintaxe do PHP – isto é, contiver erros de sintaxe – o analisador de sintaxe do PHP não será capaz de processar parte do script ou todo o script. As pessoas são boas em inferir informações a partir de dados parciais ou contraditórios. Os computadores não.

Entre muitas outras regras, a sintaxe do PHP exige que instruções terminem com ponto-e-vírgula, que strings sejam incluídas entre aspas e que os parâmetros passados às funções sejam separados por vírgulas e incluídos entre parênteses. Se desobedecermos a essas regras, nosso script de PHP provavelmente não funcionará e provavelmente gerará uma mensagem de erro na primeira vez que tentarmos executá-lo.

Uma das grandes forças do PHP são as úteis mensagens de erro que ele fornece quando algo sai errado. Uma mensagem de erro de PHP normalmente informará o que saiu errado, em que arquivo o erro ocorreu e em que linha o erro se encontra.

Uma mensagem de erro se parece com o seguinte:

```
Parse error: parse error in
/home/book/public_html/chapter25/error.php on line 2
```

Esse erro foi produzido pelo seguinte script:

```
<?php
  $date = date(m.d.y');
?>
```

Você pode ver que estamos tentando passar uma string para a função `date( )` mas deixamos acidentalmente de colocar as aspas de abertura que marcariam o começo da string.

Os erros de sintaxe simples como esse são normalmente os mais fáceis de localizar. Podemos cometer um erro parecido, mas mais difícil de localizar, esquecendo de terminar a string, como mostrado neste exemplo:

```
<?php
  $date = date('m.d.y);
?>
```

Esse script gerará a seguinte mensagem de erro:

```
Parse error: parse error in
/home/book/public_html/chapter25/error.php on line 4
```

Obviamente, como nosso script só tem três linhas, o erro não está realmente na linha quatro. Os erros em que você abre algo mas esquece de fechar freqüentemente aparecerão assim. Esse problema pode se apresentar com aspas simples e aspas duplas e também com as várias formas de parênteses.

O seguinte script gerará um erro de sintaxe semelhante:

```
<?php
  if (true)
  {
    echo 'error here';
?>
```

Esses erros podem ser difíceis de localizar se resultarem de uma combinação de diversos arquivos. Eles também podem ser difíceis de localizar se ocorrerem em um arquivo grande. Ver a mensagem parse error on line 1001 para um arquivo de 1.000 linhas pode ser o suficiente para estragar seu dia e serve como uma dica sutil de que você deve tentar escrever um código mais modular.

Mas em geral, o erro de sintaxe é o tipo de erro mais fácil de localizar. Se você cometer um erro de sintaxe, o PHP apresentará uma mensagem informando onde encontrar o equívoco.

Erros de tempo de execução

Os erros de tempo de execução podem ser difíceis de detectar e corrigir. Um script contém um erro de sintaxe ou não. Se o script contiver um erro de sintaxe, o analisador de sintaxe o detectará. Os erros de tempo de execução não são causados unicamente pelo conteúdo do script. Eles se baseiam em interações entre os scripts e outros eventos ou condições.

A seguinte instrução:

```
require ('filename.php');
```

é uma instrução de PHP perfeitamente válida. Essa instrução não contém nenhum erro de sintaxe.

Mas, essa instrução pode gerar um erro de tempo de execução. Se você executar essa instrução e `filename.php` não existir ou o usuário sob o qual o script executa tiver permissão de leitura negada, você obterá um erro semelhante a este:

```
Fatal error: Failed opening required 'filename.php'
(include_path='.:/usr/local/lib/php') in
/home/book/public_html/chapter25/error.php on line 1
```

Embora nada estivesse errado com o código, pelo fato de ele contar com um arquivo que poderia existir ou não em diferentes momentos em que o código é executado, ele pode gerar um erro de tempo de execução.

Todas as seguintes três instruções são válidas no PHP. Infelizmente, em combinação, elas tentam fazer o impossível – dividir por zero.

```
$i = 10;
$j = 0;
$k = $i/$j;
```

Esse trecho de código gerará o seguinte aviso:

```
Warning: Division by zero in
/home/book/public_html/chapter25/div0.php on line 3
```

Isso facilitará muito a correção. Poucas pessoas tentariam escrever código que tentasse dividir por zero de propósito, mas negligenciar a verificação da entrada de usuário freqüentemente resulta nesse tipo de erro.

O código a seguir às vezes gera o mesmo erro mas pode ser mais difícil de isolar e corrigir porque acontece apenas parte do tempo.

```
$i = 10
$k = $i/$_REQUEST['input'];
```

Esse é um de muitos diferentes erros de tempo de execução com que você talvez se depare ao testar o código.

Causas comuns de erros de tempo de execução incluem:

- Chamadas a funções que não existem;
- Ler ou gravar arquivos;
- Interação com MySQL ou outros bancos de dados;
- Conexões com serviços de rede;
- Falha na verificação dos dados de entrada.

Discutiremos brevemente cada um deles a seguir.

Chamadas a funções que não existem

É fácil chamar acidentalmente funções que não existem. As funções predefinidas são com freqüência inconsistentemente nomeadas. Por que `strip_tags( )` tem um sublinhado, ao passo que `stripslashes( )` não tem?

Também é fácil chamar uma de suas próprias funções que não existem no script atual, mas talvez existam em outra parte. Se o código contiver uma chamada para uma função inexistente, como:

```
nonexistent_function( );
```

ou

```
mispeled_function( );
```

você verá uma mensagem de erro semelhante a:

```
Fatal error: Call to undefined function: nonexistent_function( )
in /home/book/public_html/chapter25/error.php on line 1
```

De maneira semelhante, se você chamar uma função que existe, mas chamá-la com um número incorreto de parâmetros, receberá um aviso.

A função `strstr( )` exige duas strings: um palheiro a ser pesquisado e uma agulha a ser localizada. Se em vez disso a chamarmos assim:

```
strstr( );
```

obteremos o seguinte aviso:

```
Warning: Wrong parameter count for strstr( ) in
/home/book/public_html/chapter25/error.php on line 1
```

Essa mesma instrução dentro do seguinte script está igualmente errada:

```
<?php
  if($var == 4)
  {
    strstr( );
  }
?>
```

Exceto no caso possivelmente raro em que a variável `$var` tem o valor `4`, a chamada para `strstr( )` não ocorrerá e nenhum aviso será emitido. O interpretador do PHP não perde tempo analisando seções de código que não são necessárias para a execução atual do script. Certifique-se de que seu teste foi feito com cuidado!

É fácil chamar funções incorretamente, mas como as mensagens de erro resultantes identificam a chamada de função e linha exata que estão causando o problema, elas são igualmente fáceis de corrigir. Elas somente serão difíceis de localizar se o processo de teste for pobre e não testar todo o código condicionalmente executado. Quando você testa, um dos objetivos é executar cada linha de código exatamente uma vez. Outro objetivo é testar todas as condições de limite e classes de entrada.

Lendo ou gravando arquivos

Embora qualquer erro possa acontecer em algum ponto durante a vida útil do programa, alguns são mais prováveis que outros. Cometer um erro ao acessar arquivos é tão provável de acontecer que devemos lidar com ele elegantemente. As unidades de disco falham ou o espaço se esgota; e erro humano resulta em alteração de permissões de diretório.

Funções como `fopen( )`, que podem ocasionalmente falhar, têm em geral um valor de retorno para sinalizar que um erro ocorreu. Para `fopen( )`, um valor de retorno de `false` indica falha.

Para funções que fornecem a notificação de falha, você precisa verificar cuidadosamente o valor de retorno de cada chamada e atuar sobre as falhas.

Interação com o MySQL ou outros bancos de dados

Conectar-se ao MySQL e utilizá-lo pode gerar muitos erros. A função mysql_connect( ) sozinha pode gerar pelo menos os seguintes erros:

- **Warning:** mysqli_connect( ) [function.mysqli-connect]: Can't connect to MySQL server on 'localhost' (10061)
- **Warning:** mysqli_connect( ) [function.mysqli-connect]: Unknown MySQL Server Host 'hostname' (11001)
- **Warning:** mysqli_connect( ) [function.mysqli-connect]: Access denied for user: 'username'@'localhost' (Using password: YES)

Como você provavelmente esperaria, mysql_connect( ) fornece um valor de retorno de false quando um erro ocorre. Isso significa que você pode facilmente detectar e tratar esses tipos de erros comuns.

Se não interromper a execução normal do script e tratar esses erros, o script tentará continuar interagindo com o banco de dados. Tentar executar consultas e obter resultados sem uma conexão do MySQL válida resultará na exibição de uma tela de aparência não-profissional cheia de mensagens de erro vista por seus visitantes.

Muitas outras funções de PHP comumente utilizadas relacionadas ao MySQL como mysql_query( ) também retornam false para indicar que um erro ocorreu.

Se ocorrer um erro, você pode acessar o texto da mensagem de erro utilizando a função mysql_error( ), ou um código de erro utilizando a função mysql_errno( ). Se a última função do MySQL não gerou um erro, mysql_error( ) retorna uma string vazia e mysql_errno( ) retorna 0.

Por exemplo, assumindo que nos conectamos ao servidor e selecionamos um banco de dados para utilização, o fragmento de código

```
$result = mysql_query( 'select * from does_not_exist' );
echo mysql_errno( );
echo '<br />';
echo mysql_error( );
```

poderia gerar a saída:

```
1146
Table 'dbname.does_not_exist' doesn't exist
```

Observe que a saída dessas funções refere-se à última função do MySQL executada (diferente do mysql_error( ) ou mysql_errno( )). Se você quiser saber o resultado de um comando, certifique-se de verificá-lo antes de executar outros.

Assim como as falhas de interação de arquivo, as falhas de interação de banco de dados também ocorrerão. Mesmo depois de completar o desenvolvimento e teste de um serviço, você ocasionalmente descobrirá que o daemon do MySQL (mysqld) deixou de funcionar ou está sem conexões disponíveis. Se o banco de dados executar em outra máquina física, você conta com outro conjunto de componentes de hardware e software que poderia falhar – outra conexão de rede, placas de rede, roteadores e assim por diante entre o servidor Web e a máquina de bancos de dados.

Você precisa lembrar-se de verificar se suas solicitações de banco de dados são bem-sucedidas antes de tentar utilizar o resultado. Não há nenhum problema em tentar executar uma consulta depois de não conseguir conectar-se ao banco de dados e nenhum problema em tentar extrair e processar os resultados depois de executar uma consulta que falhou.

É importante notar a essa altura que há uma diferença entre uma consulta deficiente e uma consulta que meramente não consegue retornar quaisquer dados ou afetar qualquer linha.

Uma consulta de SQL que contém erro de sintaxe de SQL ou referencia bancos de dados, tabelas ou colunas que realmente existem poderia falhar. Por exemplo, a consulta

```
select * from does_not_exist;
```

falhará porque o nome de tabela não existe e gera um número de erro e mensagem recuperável com `mysql_errno( )` e `mysql_error( )`.

Uma consulta de SQL que é sintaticamente válida e refere-se somente aos bancos de dados, tabelas e colunas que não existem geralmente falharão. Mas a consulta pode não retornar nenhum resultado se estiver consultando uma tabela vazia ou procurando dados que não existem. Supondo que você conectou-se a um banco de dados com sucesso e tem uma tabela chamada `t1` e uma coluna chamada `c1`, a consulta:

```
select * from t1 where c1 = 'not in database';
```

será bem-sucedida mas não retornará nenhum resultado.

Antes de utilizar o resultado da consulta, você precisará verificar a ocorrência de falhas ou a possibilidade de a consulta não fornecer nenhum resultado.

Conexões para serviços de rede

Embora dispositivos e outros programas no sistema ocasionalmente falhem, eles raramente deveriam falhar a menos que sejam de má qualidade. Ao utilizar uma rede para conectar-se a outras máquinas e a software nessas máquinas, você precisará aceitar que alguma parte do sistema freqüentemente falhará. Para conectar-se de uma máquina a outra, você conta com numerosos dispositivos e serviços que não estão sob seu controle.

Com o risco de ser repetitivo, você realmente precisa verificar cuidadosamente o valor de retorno das funções que tentam interagir com um serviço de rede.

Uma chamada de função como

```
$sp = fsockopen ( 'localhost', 5000 );
```

fornecerá um aviso se falhar a conexão com a porta 5000 na máquina `localhost`, mas o exibirá no formato padrão e não dará ao script a opção de manipulá-lo de maneira elegante.

Reescrever a chamada assim

```
$sp = @fsockopen ( 'localhost', 5000, &$errorno, &$errorstr );
if(!$sp)
 echo "ERROR: $errorno: $errorstr";
```

evita a mensagem de erro, verifica o valor de retorno para ver se um erro ocorreu e utiliza o seu próprio código para lidar com a mensagem de erro. À medida que o código é escrito, exibe uma mensagem de erro que pode ajudar a resolver o problema. Nesse caso, produz a seguinte saída:

```
ERROR: 10035: A non-blocking socket operation could not be completed immediately.
```

Os erros de tempo de execução são mais difíceis de eliminar que os erros de sintaxe porque o analisador de sintaxe não pode sinalizar o erro na primeira vez em que o código é executado. Como erros de tempo de execução ocorrem em resposta a uma combinação de eventos, eles podem ser difíceis de detectar e de resolver. O analisador de sintaxe não pode informar automaticamente que uma linha particular gerará um erro. O teste precisa fornecer uma das situações que criam o erro.

Tratar erros de tempo de execução exige uma certa quantidade de reflexão anterior para verificar diferentes tipos de falha que podem ocorrer e então tomar a ação apropriada. Também exige testes cuidadosos para simular cada classe de erro de tempo de execução que possa ocorrer.

Isso não significa que você precisa tentar simular todos os diferentes erros que poderiam ocorrer. O MySQL, por exemplo, pode fornecer um entre cerca de 200 números de erro e mensagens diferentes. Você precisa simular um erro em cada chamada de função que apresente alguma chance de resultar em um erro, e um erro de cada tipo que for tratado por um bloco diferente de código.

Falha ao verificar dados de entrada

Freqüentemente fazemos suposições sobre os dados de entrada que serão inseridos por usuários. Se esses dados não atenderem às nossas expectativas, podem causar um erro, que pode ser tanto um erro de tempo de execução como um erro de lógica (detalhado na seção a seguir).

Um exemplo clássico de um erro de tempo de execução ocorre ao lidar com dados de entrada de usuário e esquecer de adicionar barras a eles com `AddSlashes( )`. Isso significa que se tivermos um usuário com um nome como O'Grady que contém um apóstrofo, obteremos um erro proveniente da função do banco de dados.

Conversaremos mais a respeito de erros por causa das suposições sobre os dados de entrada na próxima seção.

Erros de lógica

Os erros de lógica podem ser o tipo de erro mais difícil de localizar e eliminar. Esse tipo de erro é onde o código perfeitamente válido faz exatamente o que ele é instruído a fazer, mas isso não era o que o escritor pretendia.

Os erros de lógica podem ser causados por um simples erro de digitação, como:

```
for ( $i = 0; $i < 10; $i++ );
{
  echo 'doing something<br />';
}
```

Esse trecho de código é perfeitamente válido. Ele segue a sintaxe válida do PHP. Ele não conta com nenhum serviço externo, então é improvável que ele falhe em tempo de execução. A menos que o examine com muito cuidado, ele provavelmente não fará o que você imagina nem o que o programador pretendia que ele fizesse.

Num passar de olhos, parece que o código itera pelo loop `for` dez vezes, ecoando `"doing something"` a cada vez. A adição de um ponto-e-vírgula estranho no final da primeira linha significa que o loop não tem nenhum efeito nas linhas seguintes. O loop `for` iterará dez vezes sem resultado e então a instrução `echo` será executada uma vez.

Como esse código é uma maneira perfeitamente válida, porém ineficiente, de escrever código para alcançar esse resultado, o analisador de sintaxe não se queixará. Os computadores são muito bons em algumas coisas, mas eles não têm nenhum senso comum ou inteligência. Um computador fará exatamente o que foi instruído a fazer. Você precisa certificar-se de que aquilo que diz é exatamente o que deseja.

Os erros de lógica não são causados por qualquer tipo de falha do código, mas meramente por uma falha do programador em escrever código que instrui o computador a fazer exatamente o que ele quer. Como resultado, os erros não podem ser detectados automaticamente. Não lhe será dito que ocorreu um erro e você não receberá o número da linha onde está o problema. Os erros de lógica só serão detectados por meio do teste adequado.

É relativamente fácil cometer um erro de lógica como o do trivial exemplo anterior, mas também é fácil corrigir já que a primeira vez que o código executar você verá uma saída diferente da que esperava. A maioria de erros de lógica é um pouco mais insidiosa.

Normalmente, os incômodos erros de lógica resultam das suposições erradas dos desenvolvedores. O Capítulo 24 recomendou utilizar outros desenvolvedores para revisar código a fim de sugerir casos adicionais de teste e utilizar pessoas da audiência-alvo em vez de desenvolvedores para teste. É muito fácil supor que as pessoas inserirão somente certos tipos de dados e é muito fácil deixar passar erros despercebidos se você não fizer seus próprios testes.

Digamos que você tem uma caixa de texto Quantity Order em um site de comércio. Você supôs que as pessoas só inseririam números positivos? Se um visitante inserir o número dez negativo, seu software reembolsará seu cartão de crédito com dez vezes o valor do preço do item?

Suponha que você tenha uma caixa para inserir um valor monetário. Você permite que as pessoas insiram a quantidade com ou sem sinal de cifrão? Você permite que as pessoas insiram números com os milhares separados por vírgulas? Algumas desses itens podem ser verificados no lado cliente (utilizando, por exemplo, JavaScript) para tirar um pouco a carga do servidor.

Se você estiver passando informações para outra página, ocorreu a você que há caracteres que têm importância especial em um URL como espaços na string que você está passando?

Um número infinito de erros de lógica é possível. Não há maneira automatizada de verificá-los. A única solução é, primeiro, tentar eliminar suposições que você codificou implicitamente no script e, segundo, testar completamente cada possível tipo de entrada válida e inválida, assegurando que você obtenha o resultado antecipado de todas.

Auxílios para depuração de variáveis

À medida que os projetos tornam-se mais complexos, pode ser útil ter algum código utilitário para ajudar a identificar a causa de erros. Uma parte de código que você talvez considere útil está contida na Listagem 25.1. Esse código ecoará o conteúdo de variáveis passadas para a página.

Listagem 25.1 **`dump_variables.php` – Esse código pode ser incluído em páginas para fazer dump do conteúdo de variáveis para depuração**

```
<?php
  // essas linhas formatam a saída como comentários de HTML
  // e chamam dump_array repetidamente

  echo '\n<!-- BEGIN VARIABLE DUMP -->\n\n';

  echo '<!-- BEGIN GET VARS -->\n';
  echo '<!-- '.dump_array($_GET).' -->\n';

  echo '<!-- BEGIN POST VARS -->\n';
  echo '<!-- '.dump_array($_POST).' -->\n';

  echo '<!-- BEGIN SESSION VARS -->\n';
  echo '<!-- '.dump_array($_SESSION).' -->\n';

  echo '<!-- BEGIN COOKIE VARS -->\n';
  echo '<!-- '.dump_array($_COOKIE).' -->\n';

  echo '\n<!-- END VARIABLE DUMP -->\n';

// dump_array( ) usa print_r interno
// e escapa qualquer comentário HTML final

function dump_array($array)
{
 $output = print_r($array, true);
 $output = str_replace('-->', '-->, $output);
 return output;
}

?>
```

Esse código envia para a saída a quatro arrays de variáveis que uma página recebe. Se uma página foi chamada com variáveis `GET`, variáveis `POST`, cookies ou tiver variáveis de sessão, esses serão enviados para a saída.

Aqui, colocamos a saída dentro de um comentário de HTML para que seja visualizável, mas ela não interferirá na maneira como o navegador exibe os elementos visíveis da página. Essa é uma boa maneira de gerar as informações de depuração. Esconder as informações de depuração nos comentários, como na Listagem 25.1, permite que você deixe seu código depurado até o último minuto. Usamos a função `dump_array( )` para envolver `print_r( )`. A função `dump_array( )` escapa qualquer caractere de comentário HTML final.

A saída exata dependerá das variáveis passadas para a página, mas quando adicionamos um dos exemplos de autenticação do Capítulo 22 à Listagem 22.4, o código adiciona as seguintes linhas à HTML gerada pelo script:

```
<!-- BEGIN VARIABLE DUMP -->

<!-- BEGIN GET VARS -->
<!--Array
(
)
-->
<!-- BEGIN POST VARS -->
<!--Array
(
  [userid] => testuser,
  [password] => password
)
-->
<!-- BEGIN SESSION VARS -->
<!--Array
(
)
-->
<!-- BEGIN COOKIE VARS -->
<!--Array
(
   [PHPSESSID] = b2b5f56fad986dd73af33f470f3c1865
)
-->

<!-- END VARIABLE DUMP -->
```

Você pode ver que o script exibe as variáveis `POST` enviadas do formulário de login na página anterior – `userid` e `password`. Esse script também exibe a variável de sessão que estamos utilizando para manter o nome de usuário – `valid_user`. Como discutido no Capítulo 22, o PHP utiliza um cookie para vincular variáveis de sessão a usuários particulares. Nosso script está ecoando o número pseudo-aleatório, `PHPSESSID`, que é armazenado nesse cookie para identificar um usuário particular.

Níveis de informe de erros

O PHP permite configurar o quanto deve ser difícil com os erros. Você pode modificar os tipos de eventos que gerarão as mensagens. Por padrão, o PHP informará todos os erros que não sejam avisos.

O nível de informe de erro é configurado utilizando um conjunto de constantes predefinidas, mostrado na Tabela 25.1.

Tabela 25.1 **Constantes de informe de erro**

Valor	Nome	Significado
1	E_ERROR	Informa erros fatais em tempo de execução.
2	E_WARNING	Informa erros não-fatais em tempo de execução.
4	E_PARSE	Informa erros de análise sintática.

Tabela 25.1 **Continuação**

Valor	Nome	Significado
8	E_NOTICE	Informa avisos, notificações que algo feito talvez seja um erro.
16	E_CORE_ERROR	Informa falhas na inicialização do mecanismo do PHP.
32	E_CORE_WARNING	Informa falhas não-fatais durante a inicialização do mecanismo do PHP.
64	E_COMPILE_ERROR	Informa erros na compilação.
128	E_COMPILE_WARNING	Informa erros não-fatais na compilação.
256	E_USER_ERROR	Informa erros desencadeados pelo usuário.
512	E_USER_WARNING	Informa advertências desencadeadas pelo usuário.
1024	E_USER_NOTICE	Informa avisos desencadeados pelo usuário.
2047	E_ALL	Informa todos os erros e avisos.
2048	E_STRICT	Informa uso de comportamento desaprovado ou não-recomendado, não incluído em E_ALL mas muito útil para refazer código.

Cada constante representa um tipo de erro que pode ser informado ou ignorado. Se, por exemplo, você especificar o nível de erro como E_ERROR, somente os erros fatais serão informados. Essas constantes podem ser combinadas utilizando aritmética binária para produzir diferentes níveis de erro.

O nível padrão de erro, que informa todos os outros erros diferentes de avisos, é especificado como:

```
E_ALL & ~E_NOTICE
```

Essa expressão consiste em duas das constantes predefinidas combinadas utilizando operadores aritméticos de bitwise. O "e comercial" (&) é o operador de bitwise AND, e o til (~) é o operador de bitwise NOT. Essa expressão pode ser lida como E_ALL AND NOT E_NOTICE.

E_ALL por si só é efetivamente uma combinação de todos os outros tipos de erro com exceção de E_STRICT. Ele poderia ser substituído pelos outros níveis combinados juntos usando o operador de bitwise OR (|).

```
E_ERROR | E_WARNING | E_PARSE | E_NOTICE | E_CORE_ERROR | E_CORE_WARNING |
E_COMPILE_ERROR |E_COMPILE_WARNING | E_USER_ERROR | E_USER_WARNING |
E_USER_NOTICE
```

De maneira semelhante, o nível de informe de erro padrão poderia ser especificado por todos os níveis de erro exceto todos os avisos agrupados com OR.

```
E_ERROR | E_WARNING | E_PARSE | E_CORE_ERROR | E_CORE_WARNING | E_COMPILE_ERROR |
E_COMPILE_WARNING | E_USER_ERROR | E_USER_WARNING | E_USER_NOTICE
```

Alterando as configurações de informe de erro

Você pode definir as configurações de informe de erro globalmente, no arquivo php.ini ou individualmente por script.

Para alterar o informe de erro em todos os scripts, você pode modificar essas quatro linhas no arquivo php.ini padrão:

```
error_reporting =     E_ALL & ~E_NOTICE
display_errors  =     On
log_errors      =     Off
track_errors    =     Off
```

As configurações globais padrão servem para:

- Informar todos os erros exceto avisos;
- Enviar mensagens de erro como HTML para a saída padrão;
- Não gravar log de mensagens de erro em disco;
- Não monitorar erros, armazenando o erro na variável `$php_errormsg`.

A alteração mais provável que você deve fazer é elevar o nível de relatório para `E_ALL` e `E_STRICT`. Isso resultará em muitos avisos para incidentes que poderiam indicar um erro ou simplesmente resultar do fato de o programador tirar proveito da natureza fracamente tipificada do PHP e do fato de automaticamente inicializar variáveis como `0`.

Ao fazer a depuração, você poderia achar útil configurar o nível `error_reporting` mais alto. No código de produção, se você estiver fornecendo suas próprias mensagens de erro úteis, teria uma aparência mais profissional desativar `display_errors` e ativar `log_errors`, mas deixando o nível `error_reporting` alto. Você então será capaz de verificar erros detalhados nos logs se problemas forem informados.

Ativar `track_errors` pode ajudar a lidar com erros no seu próprio código, em vez de permitir que o PHP forneça sua funcionalidade padrão. Embora o PHP forneça mensagens de erro úteis, seu comportamento padrão tem uma aparência desagradável quando as coisas saem erradas.

Por padrão, quando um erro fatal ocorre, o PHP envia o seguinte para a saída

```
<br>
<b>Tipo de erro</b>:  mensagem de erro  in <b>caminho/arquivo.php</b>
on line <b>númeroDaLinha</b><br>
```

e pára de executar o script. Com erros não-fatais, o mesmo texto é enviado para a saída, mas a execução tem permissão de continuar.

Essa saída de HTML faz o erro se destacar, mas parece pobre. O estilo da mensagem de erro provavelmente não se ajustará à aparência do restante do site. Ele também poderia resultar no fato de os usuários do Netscape não verem absolutamente nenhuma saída se o conteúdo da página estiver sendo exibido dentro de uma tabela e seus navegadores estiverem cheios de HTML válida. HTML que abre mas não fecha elementos de tabela, como

```
<table>
<tr><td>
<br>
<b>Tipo de erro</b>:  mensagem de erro  in <b>caminho/arquivo.php</b>
on line <b>númeroDaLinha</b><br>
```

será exibido como uma tela em branco por alguns navegadores.

Não temos de manter o comportamento de tratamento de erro padrão do PHP, nem mesmo utilizar as mesmas configurações para todos os arquivos. Para alterar o nível de informe de erro para o script atual, você pode chamar a função `error_reporting( )`.

A passagem de uma constante de informe de erros, ou uma combinação delas, configura o nível da mesma maneira que a diretiva semelhante no `php.ini` faz. A função retorna o nível anterior do informe de erros. Uma maneira comum de utilizar a função é assim:

```
// desativa o informe de erro
$old_level = error_reporting(0);
// coloque aqui o código que gerará avisos
// ativa de novo o informe de erro
error_reporting($old_level);
```

Esse trecho de código desativará o informe de erro, permitindo executar algum código que provavelmente gerará avisos que não queremos ver.

Desativar informe de erro permanentemente é uma má idéia pois dificulta a localização e correção dos erros de codificação.

Desencadeando seus próprios erros

A função `trigger_error( )` pode ser utilizada para desencadear seus próprios erros. Os erros criados dessa maneira serão tratados da mesma maneira que erros de PHP convencionais.

A função exige uma mensagem de erro e opcionalmente pode receber um tipo de erro. O tipo de erro precisa ser um de `E_USER_ERROR`, `E_USER_WARNING` ou `E_USER_NOTICE`. Se você não especificar um tipo, o padrão é `E_USER_NOTICE`.

Você utiliza `trigger_error( )` assim:

```
trigger_error('This computer will self destruct in 15 seconds', E_USER_WARNING);
```

Tratando erros de maneira elegante

Se tiver uma base em C++ ou Java, talvez você sinta falta do tratamento de exceções ao utilizar o PHP. As exceções permitem que funções sinalizem que ocorreu um erro e permitem deixar que um handler de exceção trate do erro. As exceções foram adicionadas ao PHP apenas na versão 5, mas são uma excelente maneira de lidar com erros em grandes projetos. Elas foram adequadamente estudadas no Capítulo 7, então não as revisitaremos agora.

Se você precisa que seu código funcione no PHP4, pode simular um comportamento parecido com handlers de erros gerados pelo usuário e de erros definidos pelo usuário, mas esse comportamento se tornará menos importante agora que o PHP suporta as exceções. Você já viu que pode desencadear seus próprios erros. Você também pode fornecer seus próprios handlers de erro para capturar erros.

A função `set_error_handler( )` permite fornecer uma chamada de função quando erros de nível de usuário, avisos e notificações ocorrem. Você chama `set_error_handler( )` com o nome da função que deseja utilizar como o handler de erro.

Sua função de tratamento de erro deve aceitar dois parâmetros: um tipo de erro e uma mensagem de erro. Com base nessas duas variáveis, sua função pode decidir como tratar o erro. O tipo de erro deve ser uma das constantes de tipo de erro definida. A mensagem de erro é uma string descritiva.

Uma chamada para `set_error_handler( )` será semelhante a:

```
set_error_handler('myErrorHandler');
```

Tendo dito ao PHP para utilizar uma função chamada `myErrorHandler( )`, devemos então fornecer uma função com esse nome. Essa função deve ter o seguinte protótipo:

```
My_error_handler(int tipo_de_erro, string msg_de_erro)
                 [, string errfile [, int errline [, array errcontext]]]))
```

Mas o que ela realmente faz depende de você.

Os parâmetros passados à função de tratamento são:

- O tipo de erro;
- A mensagem de erro;
- O arquivo no qual ocorreu o erro;
- A linha na qual o erro ocorreu;
- A tabela de símbolos – ou seja, um conjunto de todas as variáveis e seus valores no momento em que ocorreu o erro.

Ações lógicas talvez incluam:

- Exibir a mensagem de erro fornecida;
- Armazenar informações em um arquivo de log;

- Enviar e-mail do erro para um endereço;
- Terminar o script com uma chamada para exit.

A Listagem 25.2 contém um script que declara um handler de erro, configura-o utilizando `set_error_handler( )` e então gera alguns erros.

Listagem 25.2 **`handle.php` – Esse script declara um handler de erro personalizado e gera erros diferentes**

```
<?php
  // A função handler de erros
  function myErrorHandler ($errno, $errstr, $errfile, $errline)
  {
    echo "<br /><table bgcolor='#cccccc'><tr><td>
          <p><b>ERROR:</b> $errstr</p>
          <p>Please try again, or contact us and tell us that
          the error occurred in line $errline of file '$errfile'</p>";
    if ($errno == E_USER_ERROR||$errno == E_ERROR)
    {
      echo '<p>This error was fatal, program ending</p>';
      echo '</td></tr></table>';
      // fecha recursos abertos, incluindo rodapé de página etc.
      exit;
    }
    echo '</td></tr></table>';
  }
  // Configura o handler de erros
  set_error_handler('myErrorHandler');

  // desencadea níveis diferentes de erro
  trigger_error('Trigger function called', E_USER_NOTICE);
  fopen('nofile', 'r');
  trigger_error('This computer is beige', E_USER_WARNING);
  include ('nofile');
  trigger_error('This computer will self destruct in 15 seconds', E_USER_ERROR);
?>
```

A saída desse script é mostrada na Figura 25.1.

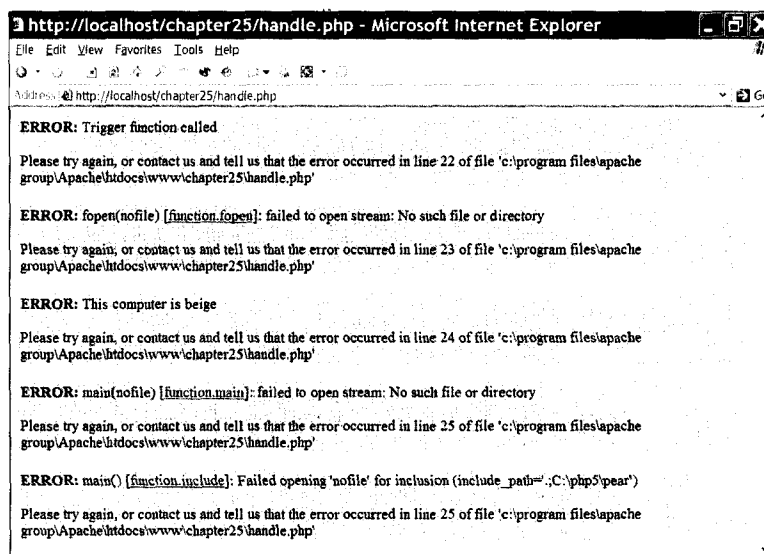



Figura 25.1 Você pode fornecer mensagens de erro mais amigáveis que o PHP se utilizar seu próprio handler de erro.

Esse handler de erro personalizado não faz nada mais que o comportamento padrão. Como esse código é escrito por você, ele pode fazer o que você quiser. Ele fornece uma escolha sobre o que dizer a seus visitantes quando algo sair errado e como apresentar essas informações para que se ajustem com o restante do site. Mais importante, ele fornece flexibilidade para decidir o que acontece. O script deve continuar? Uma mensagem deve ser registrada em log ou ser exibida? O suporte técnico deve ser alertado automaticamente?

É importante notar que seu handler de erro não terá a responsabilidade de lidar com todos os tipos de erro. Alguns erros, como erros de análise sintática e erros fatais de tempo de execução ainda desencadearão o comportamento padrão. Se isso o preocupa, certifique-se de verificar os parâmetros cuidadosamente antes de passá-los para uma função que possa gerar erros fatais; e desencadeie seu próprio nível de erro `E_USER_ERROR` se seus parâmetros forem causar falha.

Aqui está um novo recurso no PHP5: se o seu handler de erros retornar um valor `false` explícito, o handler de erros do PHP será chamado. Dessa forma, você pode manipular os erros `E_USER_*` sozinho e deixar o handler tratar os erros normais.

A seguir

No Capítulo 26, iniciaremos nosso primeiro projeto. Nesse projeto, veremos como pode você reconhecer os usuários que voltam para o site e ajustar o conteúdo apropriadamente.

26

Implementando autenticação e personalização pelo usuário

NESTE PROJETO, FAREMOS COM QUE OS USUÁRIOS registrem-se em nosso Web site. Quando eles fizerem isso, seremos capazes de monitorar tudo aquilo em que eles estão interessados e mostrar conteúdo apropriado. Isso é chamado de personalização pelo usuário.

Este projeto particular permitirá aos usuários construir um conjunto de bookmarks na Web e sugerir outros links que talvez achem interessantes com base em seu comportamento anterior. De maneira mais geral, a personalização pelo usuário pode ser utilizada em quase toda aplicação baseada na Web a fim de mostrar aos usuários o conteúdo que eles querem no formato desejado.

Neste projeto, e em outros a seguir, começaremos a examinar um conjunto de requisitos semelhantes àqueles que talvez você obtenha de um cliente. Desenvolveremos esses requisitos na forma de um conjunto de componentes de solução, construiremos um projeto para ligar esses componentes e, então, implementaremos cada um dos componentes.

Neste projeto, implementaremos as seguintes funcionalidades:

- Efetuar logon e autenticar usuários;
- Gerenciar senhas;
- Registrar preferências de usuário;
- Personalizar conteúdo;
- Recomendar conteúdo com base no conhecimento existente sobre um usuário.

O problema

Queremos construir um protótipo para um sistema de bookmarking on-line, que será chamado PHPBookmark, semelhante (mas mais limitado em funcionalidade) àquele disponível em Backflip:

http://backflip.com

Nosso sistema deve permitir que os usuários efetuem logon e armazenem seus bookmarks pessoais e obtenham recomendações para outros sites que eles talvez gostem de visitar com base em suas preferências pessoais.

Os requisitos dessa solução dividem-se em três grupos principais:

- Precisamos ser capazes de identificar os usuários individuais. Além disso, devemos ter alguma maneira de autenticá-los.
- Precisamos ser capazes de armazenar bookmarks para um usuário individual. Os usuários devem ser capazes de adicionar e excluir bookmarks.

- Precisamos ser capazes de recomendar a um usuário sites pelos quais talvez ele se interesse, com base no que já sabemos sobre ele.

Componentes da solução

Agora que conhecemos os requisitos do sistema, podemos começar a projetar a solução e seus componentes. Vamos examinar possíveis soluções para cada um dos três requisitos principais que listamos anteriormente.

Identificação do usuário e personalização

Há várias alternativas para autenticação de usuário, como vimos em outra parte deste livro. Como queremos associar um usuário a algumas informações de personalização, armazenaremos a senha e o login dos usuários em um banco de dados MySQL e o autenticaremos com base nesse banco de dados.

Se vamos permitir que os usuários efetuem logon com um nome de usuário e uma senha, precisaremos dos seguintes componentes:

- Os usuários devem ser capazes de registrar um nome de usuário e uma senha. Precisaremos de algumas restrições no comprimento e no formato do nome do usuário e da senha. Devemos armazenar senhas em um formato encriptado por razões de segurança.
- Os usuários devem ser capazes de efetuar login com os detalhes que forneceram no processo de registro.
- Os usuários devem ser capazes de efetuar logout quando tiverem concluído o uso de um site. Isso não é particularmente importante se as pessoas utilizam o site a partir do PC de casa, mas é muito importante para segurança se eles utilizam o site a partir de um PC compartilhado.
- O site precisa ser capaz de verificar se um usuário está conectado ou não, e acessar dados para um usuário conectado.
- Os usuários devem ser capazes de alterar suas senhas como uma forma de ajudar na segurança.
- Os usuários ocasionalmente esquecerão suas senhas. Eles devem ser capazes de redefinir suas senhas sem precisar de auxílio pessoal de nossa parte. Uma maneira comum de fazer isso é enviar a senha para o usuário em um endereço de e-mail indicado por ele ao se registrar. Isso significa que precisamos armazenar o endereço de e-mail do usuário quando ele se registrar. Como armazenamos as senhas em uma forma encriptada e não podemos decriptar a senha original, precisaremos realmente gerar uma nova senha, configurá-la e remetê-la para o usuário.

Para este projeto, escreveremos funções para todas essas partes da funcionalidade. A maioria delas será reutilizável, ou reutilizável com modificações menores, em outros projetos.

Armazenando bookmarks

Para armazenar bookmarks de um usuário, precisaremos configurar algum espaço em nosso banco de dados MySQL. Precisaremos da seguinte funcionalidade:

- Os usuários devem ser capazes de recuperar e visualizar o bookmark.
- Os usuários devem ser capazes de adicionar novo bookmark. Devemos verificar se esses são URLs válidos.
- Os usuários devem ser capazes de excluir bookmarks.

Novamente, podemos escrever funções para cada uma dessas funcionalidades.

Recomendando bookmarks

Poderíamos adotar diversas abordagens diferentes para recomendar bookmarks a um usuário. Poderíamos recomendar os mais populares entre todos ou os mais populares dentro de um assunto. Para este projeto, vamos implementar um sistema de sugestão baseado em "afinidades" que procura usuários que tenham um bookmark idêntico ao nosso usuário conectado e sugerir seus outros bookmarks para nosso usuário. Para evitar recomendar qualquer bookmark pessoal, só recomendaremos bookmarks armazenados por mais de um outro usuário.

Novamente podemos escrever uma função para implementar essa funcionalidade.

Visão geral da solução

Depois de alguns rabiscos no guardanapo, surgimos com o fluxograma de sistema mostrado na Figura 26.1.

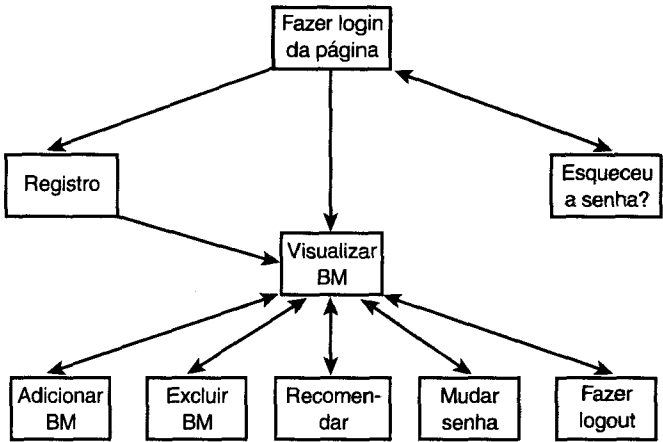



Figura 26.1 Esse diagrama mostra os possíveis caminhos por meio do sistema PHPBookmark.

Construiremos um módulo para cada caixa nesse diagrama – alguns precisarão de um script e outros, de dois. Além disso, configuraremos bibliotecas de funções para:

- Autenticação de usuário;
- Armazenamento e recuperação do bookmark;
- Validação de dados;
- Conexões de banco de dados;
- Enviar a saída para o navegador. Confinaremos toda a produção de HTML para essa biblioteca de funções, assegurando que a apresentação visual seja consistente por todo o site. (Essa é a abordagem da API de função para separar lógica e conteúdo.)

Precisaremos também construir um banco de dados de back-end para o sistema.

Descreveremos a solução detalhadamente, mas todo o código para essa aplicação pode ser encontrado no CD-ROM no diretório `chapter26`. Um resumo dos arquivos incluídos é mostrado na Tabela 26.1.

Tabela 26.1 **Arquivos na aplicação PHPBookmark**

Nome de arquivo	Descrição
bookmarks.sql	Instruções de SQL para criar o banco de dados PHPBookmark.
`login.php`	Página frontal com formulário de login para sistema.

Tabela 26.1 Continuação

Nome de arquivo	Descrição
register_form.php	Formulário para os usuários se registrarem no sistema.
register_new.php	O script para processar novos registros.
forgot_form.php	Formulário para os usuários preencherem se tiverem esquecido as senhas.
forgot_passwd.php	Script para redefinir senhas esquecidas.
member.php	Página principal de um usuário, com uma visualização de todos os seus bookmarks atuais.
add_bm_form.php	Formulário para adicionar novo bookmark.
add_bms.php	Script para realmente adicionar novos bookmarks ao banco de dados.
delete_bms.php	Script para excluir bookmarks selecionados da lista do usuário.
recommend.php	Script para sugerir recomendações para um usuário, com base em usuários com interesses semelhantes.
change_passwd_form.php	Formulário para os membros preencherem se quiserem alterar suas senhas.
change_passwd.php	Script para alterar a senha do usuário no banco de dados.
logout.php	Script para efetuar logout de um usuário da aplicação.
bookmark_fns.php	Uma coleção de includes para a aplicação.
data_valid_fns.php	Funções para validar dados de entrada de usuário.
db_fns.php	Funções para conectar-se ao banco de dados.
user_auth_fns.php	Funções para autenticação de usuário.
url_fns.php	Funções para adicionar e excluir bookmarks e fazer recomendações.
output_fns.php	Funções que formatam a saída como HTML.
bookmark.gif	Logotipo para o PHPBookmark.

Iniciaremos implementando o banco de dados MySQL para essa aplicação uma vez que ele será requerido para praticamente todas as outras funcionalidades operarem.

Então, percorreremos passo a passo o código na ordem em que ele foi escrito, iniciando na página frontal, passando pela autenticação de usuário, pelo armazenamento e pela recuperação dos bookmarks até por fim chagar a recomendação. Essa ordem é relativamente lógica – é só uma questão de elaborar as dependências e construir primeiro o que é requerido para os módulos posteriores.

Nota Para o código deste projeto funcionar da maneira como ele é escrito, você precisará ativar as aspas mágicas. Se não fizer isso, você precisará adicionar caracteres de escape com `addslashes( )` aos dados sendo inseridos ao banco de dados MySQL e eliminar caracteres de escape com `stripslashes( )` dos dados recuperados do banco de dados. Utilizamos isso como um atalho útil.

Implementando o banco de dados

O banco de dados de PHPBookmark requer apenas um esquema relativamente simples. Precisamos armazenar usuários e seus endereços de e-mail e senhas. Também precisamos armazenar o URL de um bookmark. Um usuário pode ter muitos bookmarks e muitos usuários podem registrar o mesmo bookmark. Portanto, temos duas tabelas, `user` e `bookmark`, como mostrado na Figura 26.2.

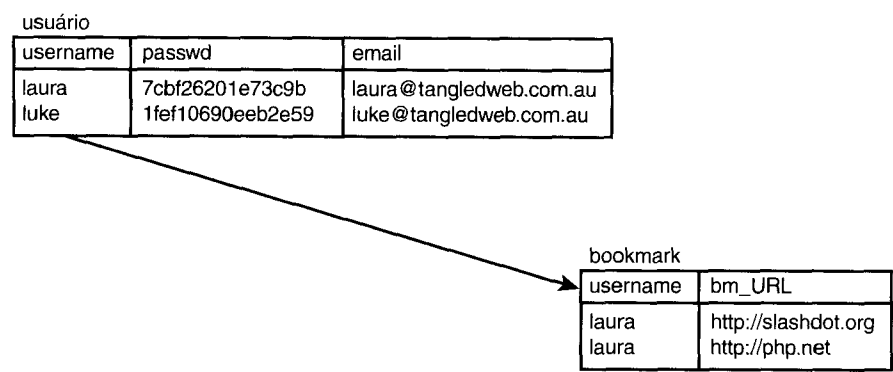



Figura 26.2 Esquema de banco de dados para o sistema de PHPBookmark.

A tabela user armazenará o nome de usuário (que é a chave primária), a senha e o endereço de e-mail do usuário. A tabela bookmark armazenará pares de nome de usuário e bookmark (bm_URL). O nome de usuário nessa tabela referenciará um nome de usuário da tabela de usuários.

A SQL para criar esse banco de dados e um usuário para conectar-se ao banco de dados Web é mostrada na Listagem 26.1. Você deve editá-la se planeja utilizá-la no sistema – altere a senha do usuário para algo mais seguro!

Listagem 26.1 **bookmarks.sql – Arquivo de SQL para configurar o banco de dados Bookmark**

```
create database bookmarks;
use bookmarks;

create table user (
  username varchar(16) not null primary key,
  passwd char(16) not null,
  email varchar(100) not null
);

create table bookmark (
  username varchar(16) not null,
  bm_URL varchar(255) not null,
  index (username),
  index (bm_URL)
 primary key(username, bm_URL)
);

grant select, insert, update, delete
on bookmarks.*
to bm_user@localhost identified by 'password';
```

Você pode configurar esse banco de dados no sistema executando esse conjunto de comandos como o usuário root do MySQL. Você pode fazer isso com o seguinte comando na linha de comando do sistema:

```
mysql -u root -p < bookmarks.sql
```

Então, aparecerá um prompt pedindo para você digitar sua senha.

Com o banco de dados configurado, vamos continuar e implementar o site básico.

Implementando o site básico

A primeira página que construiremos será chamada login.php porque ela fornece aos usuários a oportunidade de efetuar logon no sistema. O código para essa primeira página é mostrado na Listagem 26.2.

Listagem 26.2 **login.php – Página inicial do sistema de PHPBookmark**

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('');

 display_site_info( );
 display_login_form( );

 do_html_footer( );
?>
```

Esse código parece muito simples, uma vez que está principalmente chamando funções a partir da API de função que construiremos para essa aplicação. Veremos os detalhes dessas funções em um minuto. Simplesmente examinando esse arquivo, podemos ver que estamos incluindo um arquivo (contendo as funções) e então chamando algumas funções para gerar um cabeçalho HTML, exibir algum conteúdo e gerar um rodapé de HTML.

A saída desse script é mostrada na Figura 26.3.

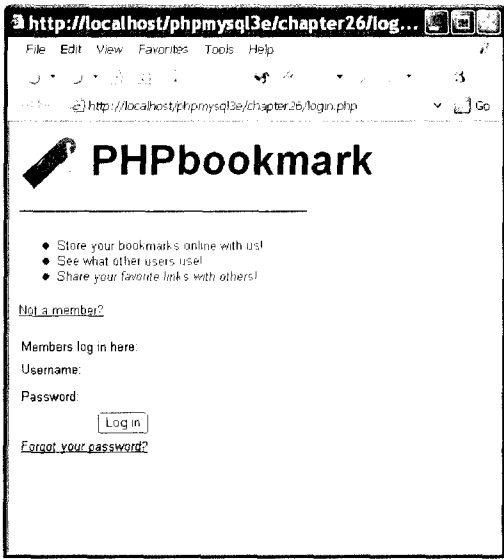



Figura 26.3 A página inicial do sistema de PHPBookmark é produzida pelas funções de renderização da HTML em `login.php`.

As funções para o sistema são inteiramente incluídas no arquivo `bookmark_fns.php`, mostrado na Listagem 26.3.

Listagem 26.3 **bookmark_fns.php – Arquivo include de funções para a aplicação Bookmark**

```
<?php
  // Podemos incluir esse arquivo em todos os nossos arquivos
  // dessa maneira, cada arquivo conterá todas as nossas funções e exceções
  require_once('data_valid_fns.php');
  require_once('db_fns.php');
  require_once('user_auth_fns.php');
  require_once('output_fns.php');
  require_once('url_fns.php');
?>
```

Como você pode ver, esse arquivo é apenas um contêiner para os cinco outros arquivos include que utilizaremos nessa aplicação. Estruturamos esse arquivo assim porque as funções dividem-se em grupos lógicos. Alguns desses grupos talvez sejam úteis para outros projetos, então colocamos cada grupo de função em um arquivo diferente em que saberemos onde localizá-los quando os quisermos novamente. Construímos o arquivo `bookmark_fns.php` porque utilizaremos a maior parte dos cinco arquivos de função na maioria de nossos scripts. É mais fácil incluir esse arquivo em cada script em vez de ter cinco instruções `require`.

Nesse caso particular, estamos utilizando funções provenientes do arquivo `output_fns.php`. Todas são funções simples e diretas que geram saída relativamente simples em HTML. Esse arquivo inclui as quatro funções que utilizamos no `login.php`, isto é, `do_html_header( )`, `display_site_info( )`, `display_login_form( )` e `do_html_footer( )`, entre outras.

Não examinaremos todas essas funções em detalhe, mas veremos uma como exemplo. O código para `do_html_header( )` é mostrado na Listagem 26.4.

Listagem 26.4 **Função `do_html_header( )` proveniente de `output_fns.php` – Essa função gera o cabeçalho padrão que aparecerá em cada página da aplicação**

```
function do_html_header($title)
{
  // imprime um cabeçalho de HTML
?>
  <html>
  <head>
    <title><?php echo $title;?></title>
    <style>
      body { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
      li, td { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
      hr { color: #3333cc; width=300; text-align=left}
      a { color: #000000 }
    </style>
  </head>
  <body>
  <img src="bookmark.gif" alt="PHPbookmark logo" border="0"
       align="left" valign="bottom" height="55" width="57" />
  <h1> PHPbookmark</h1>
  <hr />
<?php
  if($title)
    do_html_heading($title);
}
```

Como você pode ver, a única lógica nessa função é adicionar o título e o cabeçalho apropriado à página. As outras funções que utilizamos em `login.php` são semelhantes. A função `display_site_info( )` adiciona algum texto geral ao site; `display_login_form( )` exibe o formulário acinzentado mostrado na Figura 26.3; e `do_html_footer( )` adiciona um rodapé HTML padrão à página.

As vantagens de isolar ou remover HTML do principal fluxo lógico são discutidas no Capítulo 24. Utilizaremos a abordagem da API de função aqui.

Examinando a Figura 26.3, você pode ver que há três opções nessa página – os usuários podem registrar-se, efetuar logon se já se registraram ou redefinir sua senha se a esqueceram. Para implementar esses módulos mudaremos para a próxima seção, autenticação de usuário.

Implementando autenticação de usuário

Há quatro elementos principais para o módulo de autenticação de usuário: registro de usuário, login e logout, alterar senhas e redefinir senhas. Veremos cada um desses individualmente.

Registrando-se

Para registrar um usuário, precisamos obter os detalhes via um formulário e inseri-lo no banco de dados.

Quando um usuário clica no link "Not a member?" na página `login.php`, ele é direcionado para um formulário de registro produzido por `register_form.php`. Esse script é mostrado na Listagem 26.5.

Listagem 26.5 **`register_form.php` – Esse formulário fornece a oportunidade aos usuários de registrarem-se com o PHPBookmarks**

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('User Registration');

 display_registration_form( );

 do_html_footer( );
?>
```

Novamente, você pode ver que essa página é relativamente simples e apenas chama funções provenientes da biblioteca de saída no arquivo `output_fns.php`. A saída desse script é mostrada na Figura 26.4.

O formulário acinzentado nessa página é gerado pela função `display_registration_form( )`, contida em `output_fns.php`. Ao clicar no botão Register, o usuário é levado para o script `register_new.php`. Esse script é mostrado na Listagem 26.6.

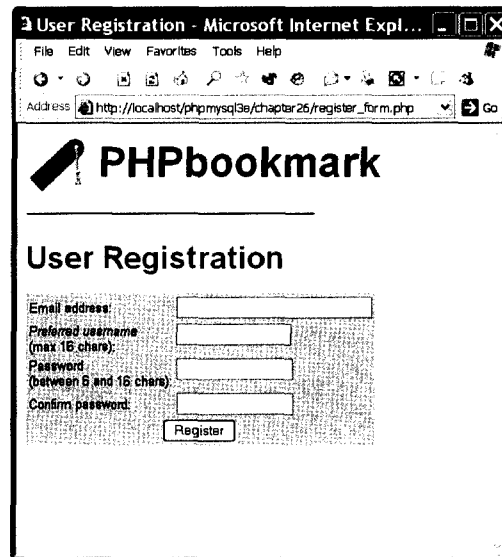



Figura 26.4 O formulário de registro recupera os detalhes de que precisamos para o banco de dados. Fazemos os usuários digitarem sua senha duas vezes, no caso de eles cometerem algum erro.

Listagem 26.6 **`register_new.php` – Esse script valida os dados do novo usuário e os coloca no banco de dados**

```
<?php
 // inclui arquivos de função para essa aplicação
 require_once('bookmark_fns.php');
```

Listagem 26.6 **Continuação**

```
//cria nomes de variáveis abreviados
$email=$_POST['email'];
$username=$_POST['username'];
$passwd=$_POST['passwd'];
$passwd2=$_POST['passwd2'];
// inicia sessão que pode ser necessária mais tarde
// inicia agora porque deve entrar antes dos cabeçalhos
session_start( );
try
{
  // verifica os formulários preenchidos
  if (!filled_out($_POST))
  {
    throw new Exception('You have not filled the form out correctly '
                     .'- please go back and try again.');
  }

  // endereço de e-mail não é válido
  if (!valid_email($email))
  {
    throw new Exception('That is not a valid email address.  Please go back '
                     .' and try again.');
  }

  // senhas não são as mesmas
  if ($passwd != $passwd2)
  {
    throw new Exception('The passwords you entered do not match
                     .' - please go back and try again.');
  }

  // verificação do cumprimento da senha está ok
  if (strlen($passwd)<6)
  {
    throw new Exception('Your password must be at least 6 characters long.'
                     .'Please go back and try again.');
  }

  // verificação do cumprimento do nome de usuário está ok
  if (strlen($username)>16)
  {
    throw new Exception('Your username must be less than 17 characters long.'
                     .'Please go back and try again.');
  }

  // tenta registrar
  // essa função também pode lançar uma exceção
  register($username, $email, $passwd);
  // registra variável de sessão
  $_SESSION['valid_user'] = $username;

  // fornece link para a página dos membros
  do_html_header('Registration successful');
  echo 'Your registration was successful.  Go to the members page '
       .'to start setting up your bookmarks!';
  do_html_url('member.php', 'Go to members page');

 // final da página
 do_html_footer( );
}
```

Listagem 26.6 Continuação

```
catch (Exception $e)
{
   do_html_header('Problem:');
   echo $e->getMessage( );
   do_html_footer( );
   exit;
}
?>
```

Esse é o primeiro script sem complexidade que vimos nessa aplicação. O script começa incluindo os arquivos de função da aplicação e iniciando uma sessão. (Quando o usuário for registrado, criaremos seu nome de usuário como uma variável de sessão da mesma maneira que fizemos no Capítulo 22.)

O corpo do script está localizado no bloco `try` porque você verifica várias condições. Se nenhuma delas falhar, a execução abortará para o bloco `catch`, que veremos em breve.

Em seguida, validamos os dados de entrada do usuário. Há várias condições que devemos testar. Essas variáveis são:

- Verificar se o formulário foi preenchido. Testamos isso com uma chamada à função `filled_out( )` da seguinte maneira:

  ```
  if (!filled_out($_POST))
  ```

  Essa função é uma função que nós mesmos escrevemos. Ela está na biblioteca de funções no arquivo `data_valid_fns.php`. Veremos essa função em um minuto.

- Verificar se o endereço de e-mail fornecido é válido. Testamos isso da seguinte maneira:

  ```
  if (valid_email($email))
  ```

  Novamente, essa é uma função que escrevemos, que está nos biblioteca `data_valid_fns.php`.

- Verificar se as duas senhas que o usuário sugeriu são as mesmas, da seguinte maneira:

  ```
  if ($passwd != $passwd2)
  ```

- Verificar se a senha tem o comprimento apropriado, como segue:

  ```
  if (strlen($passwd)<6)
  ```

  e

  ```
  if (strlen($passwd) >16)
  ```

  No exemplo, a senha deve ter pelo menos seis caracteres para dificultar mais a adivinhação e menos de 16 caracteres, para ajustar-se ao banco de dados. Observe que o comprimento máximo da senha não é restrito dessa forma porque está armazenado no hash SHA1, que sempre terá 40 caracteres de comprimento, independente do tamanho da senha.

As funções de validação de dados que utilizamos aqui, `filled_out( )` e `valid_email( )`, são mostradas nas Listagens 26.7 e 26.8, respectivamente.

Listagem 26.7 **Função `filled_out( )` de `data_valid_fns.php` – Essa função verifica se o formulário foi preenchido**

```
function filled_out($form_vars)
{
  // testa se cada variável tem um valor
  foreach ($form_vars as $key => $value)
```

Listagem 26.7 Continuação

```
  {
     if (!isset($key) || ($value == ''))
        return false;
  }
  return true;
}
```

Listagem 26.8 **Função valid_email( ) de data_valid_fns.php – Essa função verifica se um endereço de e-mail é válido**

```
function valid_email($address)
{
  // verifica se um endereço de e-mail é possivelmente válido
  if (ereg('^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$', $address))
    return true;
  else
    return false;
}
```

A função filled_out( ) espera que seja passado um array de variáveis – em geral, esse array será o $_POST ou $_GET. Ela verificará se todos eles foram preenchidos e retornará true se foram; e false, caso contrário.

A função valid_email( ) utiliza a expressão regular que desenvolvemos no Capítulo 4, para validar endereços de e-mail. Ela retorna true se um endereço parecer válido; e false, se não.

Depois que validarmos os dados de entrada, podemos realmente tentar e registrar o usuário. Se você olhar outra vez a Listagem 26.6, verá que fazemos isso da seguinte maneira:

```
   register($username, $email, $passwd);
   // registra variável de sessão
   $_SESSION['valid_user'] = $username;

   // fornece link para a página dos membros
   do_html_header('Registration successful');
   echo 'Your registration was successful.  Go to the members page '
        .'to start setting up your bookmarks!';
   do_html_url('member.php', 'Go to members page');

  // final da página
  do_html_footer( );
```

Como você pode ver, estamos chamando a função register( ) com o nome de usuário, o endereço de e-mail e a senha que foram inseridos. Se isso for bem-sucedido, registramos o nome de usuário como uma variável de sessão e fornecemos ao usuário um link para a página de membros principal. Essa é a saída mostrada na Figura 26.5.

A função register( ) está na biblioteca incluída denominada user_auth_fns.php. Essa função é mostrada na Listagem 26.9.

Listagem 26.9 **Função register( ) de user_auth_fns.php – Essa função tenta colocar informações do novo usuário no banco de dados**

```
 function register($username, $email, $password)
 // registra nova pessoa com db
 // retorna true ou mensagem de erro
 {
  // conecta ao db
```

Listagem 26.9 **Continuação**

```
$conn = db_connect( );

// verifica de nome do usuário é único
$result = $conn->query("select * from user where username='$username'");
if (!$result)
  throw new Exception('Could not execute query');
if ($result->num_rows>0)
  throw new Exception('That username is taken '
                      .'- go back and choose another one.') ;

// if ok, put in db
$result = $conn->query("insert into user values
                      ('$username', sha1('$password'), '$email')");
if (!$result)
  throw new Exception('Could not register you in database '
                      .'- please try again later.');

return true;
}
```

Figura 26.5 O registro foi bem-sucedido – agora o usuário pode ir para a página de membros.

Não há nada particularmente novo nessa função – ela conecta-se ao banco de dados que configuramos anteriormente. Se o nome de usuário selecionado já existir no banco de dados ou se o banco de dados não puder ser atualizado, a função retornará `false`. Caso contrário, ela atualizará o banco de dados e retornará `true`.

Algo a ser notado é que estamos realizando a conexão real de banco de dados com uma função que escrevemos, chamada `db_connect( )`. Essa função fornece simplesmente uma única localização que contém o nome de usuário e a senha para conectar-se ao banco de dados. Dessa maneira, se alteramos a senha de banco de dados, só precisamos alterar um arquivo na nossa aplicação. A função é mostrada na Listagem 26.10.

Listagem 26.10 **Função db_connect( ) de db_fns.php – Essa função conecta-se ao banco de dados MySQL**

```
function db_connect( )
{
  $result = new mysqli('localhost', 'bm_user', 'password', 'bookmarks');
  if (!$result)
    throw new Exception('Could not connect to database server');
  else
    return $result;
}
```

Quando os usuários estão registrados, eles podem efetuar logon e logout utilizando páginas de login e logout convencionais. Construiremos essas páginas em seguida.

Efetuando logon

Se os usuários digitarem seus detalhes no formulário em `login.php` (veja a Figura 26.3) e enviá-lo, serão direcionados ao script chamado `member.php`. Esse script fará com que eles efetuem o login se vierem desse formulário. Ele também exibirá qualquer bookmark relevante para usuários que estiverem conectados. Ele é o centro do restante da aplicação. Esse script é mostrado na Listagem 26.11.

Listagem 26.11 **member.php – Esse script é o núcleo da aplicação**

```
<?php

// inclui arquivos de função para esta aplicação
require_once('bookmark_fns.php');
session_start( );

//cria nomes de variáveis abreviados
$username = $_POST['username'];
$passwd = $_POST['passwd'];

if ($username && $passwd)
// acabaram de tentar efetuar log in
{
 try
 {
   login($username, $passwd);
   // se estiverem no banco de dados, registra o id do usuário
   $_SESSION['valid_user'] = $username;
 }
 catch(Exception $e)
 {
   // login malsucedido
   do_html_header('Problem:');
   echo 'You could not be logged in.
         You must be logged in to view this page.';
   do_html_url('login.php', 'Login');
   do_html_footer( );
   exit;
 }
}

do_html_header('Home');
check_valid_user( );
// obtém o bookmark que esse usuário salvou
```

Listagem 26.11 Continuação

```
if ($url_array = get_user_urls($_SESSION['valid_user']))
 display_user_urls($url_array);

// fornece o menu de opções
display_user_menu( );

do_html_footer( );
?>
```

Talvez você reconheça a lógica nesse script: estamos reutilizando algumas idéias do Capítulo 22.

Primeiro, verificamos se o usuário veio da página inicial – isto é, se ele acabou de preencher o formulário de login – e tentamos efetuar seu logon da seguinte maneira:

```
 if ($username && $passwd)
// acabaram de tentar efetuar log in
{
 try
 {
   login($username, $passwd);
   // se estiverem no banco de dados, registra o id do usuário

   $_SESSION['valid_user'] = $username;
 }
```

Você pode ver que estamos tentando efetuar logon utilizando uma função chamada `login( )`. Definimos isso na biblioteca `user_auth_fns.php` e veremos o código para isso em um minuto.

Se o usuário efetuou login com sucesso, registramos sua sessão como fizemos antes, armazenando o nome de usuário na variável de sessão `valid_user`.

Se tudo correu bem, então mostramos ao usuário a página de membros:

```
do_html_header('Home');
check_valid_user( );
// obtém o bookmark que esse usuário salvou
if ($url_array = get_user_urls($HTTP_SESSION_VARS['valid_user']));
  display_user_urls($url_array);

// fornece o menu de opções
display_user_menu( );

do_html_footer( );
```

Novamente, essa página é composta utilizando as funções de saída. Você notará que estamos utilizando várias outras novas funções. São elas: `check_valid_user( )`, de `user_auth_fns.php`; `get_user_urls( )`, de `url_fns.php`; e `display_user_urls( )`, de `output_fns.php`. A função `check_valid_user( )` verifica se o usuário atual possui uma sessão registrada. Essa se destina *não* aos usuários que acabaram de efetuar logon, mas aos que estão no meio de uma sessão. A função `get_user_urls( )` obtém os bookmarks de um usuário a partir do banco de dados, e `display_user_urls( )` imprime os bookmarks no navegador dentro de uma tabela. Veremos `check_valid_user( )` em um instante e as outras duas na seção sobre armazenamento e recuperação de bookmarks.

O script `member.php` finaliza a página exibindo um menu com a função `display_user_menu( )`.

Uma saída de exemplo como é exibida por `member.php` é mostrada na Figura 26.6.

Agora veremos as funções `login( )` e `check_valid_user( )` um pouco mais de perto. A função `login( )` é mostrada na Listagem 26.12.

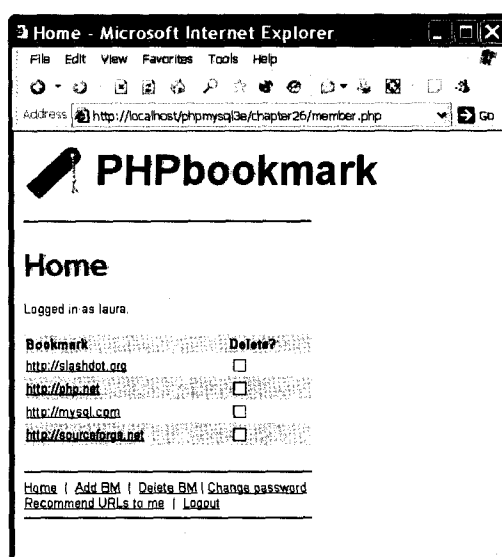



Figura 26.6 O script member.php verifica se um usuário está conectado, recupera e exibe seus bookmarks, e fornece um menu de opções.

Listagem 26.12 **Função login( ) a partir de user_auth_fns.php – Essa função verifica informações pessoais do usuário contra o banco de dados**

```
function login($username, $password)
// verifica nome de usuário e senha com db
// se coincide, returna true
// caso contrário, lança exceção
{
 // conecta ao db
 $conn = db_connect( );

 // verifica se nome de usuário é único
 $result = $conn->query("select * from user
                        where username='$username'
                        and passwd = sha1('$password')");
 if (!$result)
    throw new Exception('Could not log you in.');

 if ($result->num_rows>0)
    return true;
 else
    throw new Exception('Could not log you in.');
}
```

Como você pode ver, essa função conecta-se ao banco de dados e verifica se há um usuário com a combinação de nome de usuário e senha fornecida. Ela retornará true se houver isso ou lançará um exceção se não houver ou se as credenciais do usuário não puderem ser verificadas.

A função check_valid_user( ) não se conecta ao banco de dados novamente, mas em vez disso só verifica se o usuário tem uma sessão registrada, isto é, se ele efetuou logon. Essa função é mostrada na Listagem 26.13.

Listagem 26.13 **Função check_valid_user( ) de user_auth_fns.php – Essa função verifica se o usuário tem uma sessão válida**

```
function check_valid_user( )
// verifica de alguém efetuou log in e notifica-os se não
```

Listagem 26.13 **Continuação**

```
 if (isset($_SESSION['valid_user']))
 {
     echo 'Logged in as '.stripslashes($_SESSION['valid_user']).'.';
     echo '<br />';
 }
 else
 {
    // não efetuaram log in
    do_html_heading('Problem:');
    echo 'You are not logged in.<br />';
    do_html_url('login.php', 'Login');
    do_html_footer( );
    exit;
 }
}
```

Se o usuário não efetuou login, a função lhe dirá que ele tem de estar conectado para ver essa página e oferecerá um link para a página de login.

Efetuando logout

Talvez você tenha notado que há um link rotulado "Logout" no menu na Figura 26.6. Esse é um link para o script `logout.php`. O código para esse script é mostrado na Listagem 26.14.

Listagem 26.14 **`logout.php` – Esse script termina uma sessão de usuário**

```
<?php

// inclui arquivos de função para essa aplicação
require_once('bookmark_fns.php');
session_start( );
$old_user = $_SESSION['valid_user'];
// armazena para testar se eles *estavam* conectados
unset($_SESSION);
$result_dest = session_destroy( );

// começa a gerar a html
do_html_header('Logging Out');

if (!empty($old_user))
{
  if ($result_dest)
  {
    // se os usuários estavam conectados e agora estão desconectados
    echo 'Logged out.<br />';
    do_html_url('login.php', 'Login');
  }
  else
  {
   // se os usuários estavam conectados e não puderam ser desconectados
    echo 'Could not log you out.<br />';
  }
}
else
{
  // se eles não estavam conectados mas vieram para esta página de alguma maneira
  echo 'You were not logged in, and so have not been logged out.<br />';
```

Listagem 26.14 Continuação

```
  do_html_url('login.php', 'Login');
}

do_html_footer( );

?>
```

Novamente, talvez você ache que esse código parece familiar. Isso porque ele é baseado no código que escrevemos no Capítulo 22.

Alterando senha

Se um usuário seguir a opção de menu "Change Password", ele receberá o formulário mostrado na Figura 26.7.

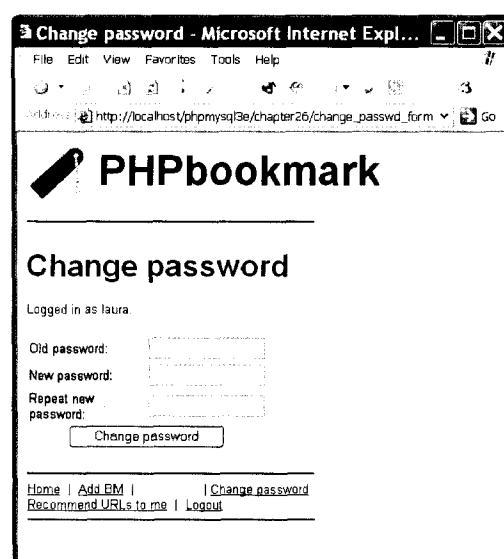



Figura 26.7 O script change_passwd_form.php fornece um formulário no qual os usuários podem alterar suas senhas.

Esse formulário é gerado pelo script change_passwd_form.php. Esse é um script simples que utiliza somente as funções provenientes da biblioteca de saída; portanto, não incluímos o código-fonte dele aqui.

Quando esse formulário é enviado, ele desencadeia o script change_passwd.php, que é mostrado na Listagem 26.15.

Listagem 26.15 **change_passwd.php** – Esse script tenta alterar a senha de um usuário

```
<?php
 require_once('bookmark_fns.php');
 session_start( );
 do_html_header('Changing password');

 // cria nomes de variável abreviados
 $old_passwd = $_POST['old_passwd'];
 $new_passwd = $_POST['new_passwd'];
 $new_passwd2 = $_POST['new_passwd2'];
```

Listagem 26.15 **Continuação**

```
try
{
  check_valid_user( );
  if (!filled_out($_POST))
    throw new Exception('You have not filled out the form completely.'
                        .' Please try again.');
  if ($new_passwd!=$new_passwd2)
     throw new Exception('Passwords entered were not the same.  Not changed.');
  if (strlen($new_passwd)<6)
     throw new Exception('New password must be at least 6 characters.'
                         .' Try again.');
  // tenta atualizar
  change_password($_SESSION['valid_user'], $old_passwd, $new_passwd);
  echo 'Password changed.';
}
catch (Exception $e)
{
  echo $e->getMessage( );
}
display_user_menu( );
do_html_footer( );
?>
```

Esse script verifica se o usuário efetuou logon (utilizando `check_valid_user( )`), se ele preencheu o formulário de senha (utilizando `filled_out( )`) e se as novas senhas são as mesmas e têm o comprimento certo. Nada disso é novo. Se tudo correr bem, ele chamará a função `change_password( )` da seguinte maneira:

```
change_password($_SESSION['valid_user'], $old_passwd, $new_passwd);
echo 'Password changed.';
```

Essa função é proveniente de nossa biblioteca `user_auth_fns.php` e o código para essa função é mostrado na Listagem 26.16.

Listagem 26.16 **Função `change_password( )` de `user_auth_fns.php` – Essa função tenta atualizar a senha de um usuário no banco de dados**

```
function change_password($username, $old_password, $new_password)
// muda senha do usuário de /old_password para new_password
// returna true ou false
{
 // se old password estiver certa
 // muda a senha para new_password e returna true
 // se não, lança uma exceção
 login($username, $old_password);
 $conn = db_connect( );
 $result = $conn->query( "update user
                          set passwd = sha1('$new_password')
                          where username = '$username'");
 if (!$result)
   throw new Exception('Password could not be changed.');
 else
   return true;  // modificação bem-sucedida
}
```

Essa função verifica se a senha antiga fornecida está correta, utilizando a função `login( )` que já vimos. Se ela for correta, então a função conecta-se ao banco de dados e atualiza a senha para o novo valor.

Redefinindo senhas esquecidas

Além de alterar senhas, precisamos lidar com a situação comum em que um usuário esqueceu sua senha. Observe que na página inicial, login.php, fornecemos um link para usuários nessa situação, rotulada como Forgotten your password? (Esqueceu sua senha?). Esse link levará usuários para o script chamado forgot_form.php, que utiliza as funções de saída para exibir um formulário conforme mostrado na Figura 26.8.

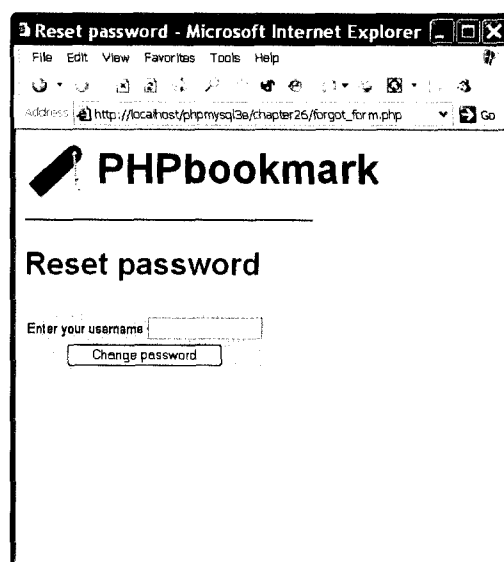



Figura 26.8 O script forgot_form.php fornece um formulário em que usuários podem pedir que suas senhas sejam redefinidas e enviadas para eles.

Esse script é muito simples – apenas utilizando as funções de saída – então não o analisaremos aqui. Quando o formulário é enviado, ele chama o script forgot_passwd.php, que é mais interessante. Esse script é mostrado na Listagem 26.17.

Listagem 26.17 **forgot_passwd.php – Esse script redefine a senha de um usuário para um valor aleatório e envia a nova por e-mail para ele**

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('Resetting password');

 // criando um nome de variável abreviado
 $username = $_POST['username'];

 try
 {
   $password = reset_password($username);
   notify_password($username, $password);
   echo 'Your new password has been emailed to you.<br />';
 }
 catch (Exception $e)
 {
   echo 'Your password could not be reset - please try again later.';
 }
 do_html_url('login.php', 'Login');
 do_html_footer( );
?>
```

Como você pode ver, esse script utiliza duas funções principais para fazer seu trabalho: reset_password( ) e notify_password( ). Vejamos uma de cada vez.

A função reset_password( ) gera uma senha aleatória para o usuário e coloca no banco de dados. O código para essa função é mostrado na Listagem 26.18.

Listagem 26.18 **Função reset_password( ) proveniente de user_auth_fns.php – Esse script redefine a senha de um usuário para um valor aleatório e envia para ele a nova senha por e-mail**

```
function reset_password($username)
// configura uma senha de valor aleatório para o nome de usuário
// retorna nova senha ou false em caso de falha
{
 // obtém uma palavra de dicionário aleatória medindo entre 6 e 13 chars
 $new_password = get_random_word(6, 13);

 if($new_password==false)
   throw new Exception('Could not generate new password.');
 // acrescenta um número ente 0 e 999 a ela
 // para transformá-la em uma senha um pouco melhor
 srand ((double) microtime( ) * 1000000);
 $rand_number = rand(0, 999);
 $new_password .= $rand_number;

 // configura a senha do usuário para isso em um banco de dados ou retorna false
 $conn = db_connect( );
 $result = $conn->query( "update user
                         set passwd = sha1('$new_password')
                         where username = '$username'");
 if (!$result)
   throw new Exception('Could not change password.');  // not changed
 else
   return $new_password;  // modificação bem-sucedida
}
```

Essa função gera sua senha aleatória obtendo uma palavra aleatória de um dicionário, utilizando a função get_random_word( ) e colocando um sufixo com um número aleatório entre 0 e 999. A função get_random_word( ) também está na biblioteca user_auth_fns.php. Essa função é mostrada na Listagem 26.19.

Listagem 26.19 **Função get_random_word( ) de user_auth_fns.php – Essa função obtém uma palavra aleatória do dicionário para utilizar na geração de senhas**

```
function get_random_word($min_length, $max_length)
// obtém uma palavra aleatória de dicionário entre os dois comprimentos
// e a retorna
{
   // gera uma palavra aleatória
  $word = '';
  //lembre-se de alterar esse caminho para adequar ao seu sistema
  $dictionary = '/usr/dict/words';  // o dicionário ispell
  $fp = fopen($dictionary, 'r');
  if(!$fp)
    return false;
  $size = filesize($dictionary);

  // vai para uma localização aleatória no dicionário
  srand ((double) microtime( ) * 1000000);
  $rand_location = rand(0, $size);
```

Listagem 26.19 Continuação

```
   fseek($fp, $rand_location);

   // obtém a próxima palavra inteira com o comprimento certo no arquivo
   while (strlen($word)< $min_length || strlen($word)>$max_length
                                      || strstr($word, "'"))
   {
      if (feof($fp))
         fseek($fp, 0);        // se alcançar o final, volta para o início
      $word = fgets($fp, 80);  // pula a primeira palavra já que poderia ser parcial
      $word = fgets($fp, 80);  // a senha potencial
   };
   $word=trim($word); // elimina o \n final de fgets
   return $word;
}
```

Para funcionar, essa função precisa de um dicionário. Se você estiver utilizando um sistema UNIX, o verificador ortográfico ispell integrado vem com um dicionário de palavras, em geral localizado em /usr/dict/words, como aqui, ou em /usr/share/dict/words. Se você não encontrá-lo em um desses lugares, na maioria dos sistemas você poderá encontrar o seu digitando:

```
$ locate dict/words
```

Se estiver utilizando algum outro sistema ou não quiser instalar o ispell, não se preocupe! Você pode fazer download de listas de palavra no formato utilizado pelo ispell em:

http://wordlist.sourceforge.net/

Esse site também tem dicionários em muitos outros idiomas, então se quiser uma palavra aleatória, digamos, norueguesa ou do esperanto, você pode fazer download de um desses dicionários. Esses arquivos são formatados com cada palavra em uma linha separada.

Para obter uma palavra aleatória desse arquivo, selecionamos uma localização aleatória entre 0 e o tamanho do arquivo e fazemos uma leitura do arquivo aí. Se lermos a partir da localização aleatória para a próxima nova linha, provavelmente vamos obter somente uma palavra parcial, então pulamos a linha em que abrimos o arquivo e tomamos a próxima palavra como nossa chamando fgets( ) duas vezes.

A função tem dois aspectos inteligentes. O primeiro é que, se alcançarmos o final do arquivo enquanto procuramos uma palavra, voltamos para o início:

```
if (feof($fp))
  fseek($fp, 0);          // se alcançar o final, volta para o início
```

O segundo é que podemos procurar uma palavra de um comprimento particular – verificamos cada palavra que puxamos do dicionário, e, se não estiver entre $min_length e $max_length, continuamos pesquisando. Ao mesmo tempo, também descartamos palavras com apóstrofos (aspas simples). Poderíamos "escapar" esse sinal ao utilizar a palavra, mas é mais fácil simplesmente obter a próxima palavra.

De volta a reset_password( ), depois de gerarmos uma nova senha, atualizamos o banco de dados para refletir isso e retornamos a nova senha de volta para o script principal. Isso então será passado para notify_password( ), que vai enviá-lo por e-mail para o usuário.

Examinemos a função notify_password( ), mostrada na Listagem 26.20.

Listagem 26.20 **Função notify_password( ) de user_auth_fns.php – Essa função envia por e-mail uma senha redefinida para um usuário**

```
function notify_password($username, $password)
// notifica o usuário que a senha foi alterada
{
   $conn = db_connect( );
```

Listagem 26.20 Continuação

```
  $result = $conn->query("select email from user
                          where username='$username'");
  if (!$result)
  {
    throw new Exception('Could not find email address.');
  }
  else if ($result->num_rows==0)
  {
    throw new Exception('Could not find '
                       .' email address.');   // nome de usuário não está em db
  }
  else
  {
    $row = $result->fetch_object( );
    $email = $row->email;
    $from = "From: support@phpbookmark \r\n";
    $mesg = "Your PHPbookmark password has been changed to $password \r\n"
            ."Please change it next time you log in. \r\n";


    if (mail($email, 'PHPbookmark login information', $mesg, $from))
      return true;
    else
      throw new Exception('Could not send email.');
  }
}
```

Nessa função, dados um nome de usuário e uma nova senha, simplesmente pesquisamos o endereço de e-mail desse usuário no banco de dados e utilizamos a função `mail( )` do PHP para enviá-la para ele.

Seria mais seguro fornecer aos usuários uma senha verdadeiramente aleatória – feita de qualquer combinação de letras minúsculas e maiúsculas, números e pontuação – em vez de um número e uma palavra aleatórios. Entretanto, uma senha como `'zigzag487'` facilitará mais a leitura e a digitação do usuário do que uma senha verdadeiramente aleatória. Freqüentemente, é confuso para usuários determinar se um caractere em uma string aleatória é 0 ou O (zero ou a letra maiúscula O); 1 ou l (o numeral um ou a letra L minúscula).

No nosso sistema, o arquivo de dicionário contém aproximadamente 45.000 palavras. Se um cracker soubesse como estamos criando senhas e soubesse o nome de um usuário, ele ainda teria de tentar 22.500.000 senhas em média para adivinhar uma. Esse nível de segurança parece adequado a esse tipo de aplicação mesmo se nossos usuários negligenciarem o conselho enviado por e-mail para alterá-la.

Implementando armazenamento e recuperação de bookmarks

Agora continuaremos e examinaremos a maneira como os bookmarks de um usuário são armazenados, recuperados e excluídos.

Adicionando bookmarks

Os usuários podem adicionar bookmarks clicando no link `Add BM` no menu de usuário. Isso os levará para o formulário mostrado na Figura 26.9.

Novamente, esse script é simples e utiliza somente as funções de saída, então não o examinaremos em detalhes aqui. Quando o formulário é enviado, ele chama o script `add_bms.php`, que é mostrado na Listagem 26.21.

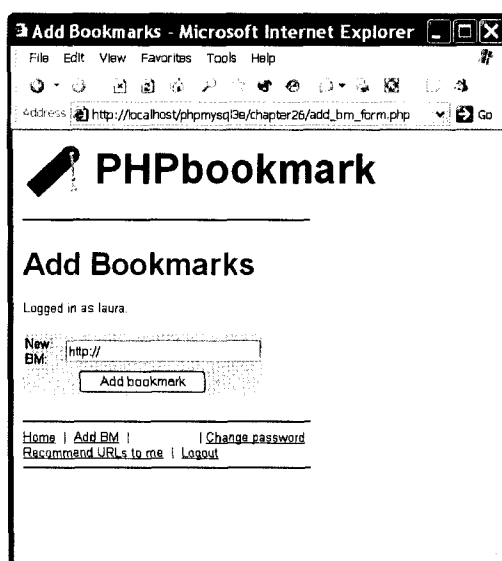



Figura 26.9 O script `add_bm_form.php` fornece um formulário no qual usuários podem adicionar bookmarks às suas páginas de bookmark.

Listagem 26.21 **add_bms.php** – Esse script adiciona novos bookmarks à página pessoal de um usuário

```
<?php
require_once('bookmark_fns.php');
session_start( );

 //cria nome de variável abreviado
 $new_url = $_POST['new_url'];

 do_html_header('Adding bookmarks');

 try
 {
   check_valid_user( );
   if (!filled_out($_POST))
   {
     throw new Exception('Form not completely filled out.');
   }
   // verifica formato de URL
   if (strstr($new_url, 'http://')===false)
      $new_url = 'http://'.$new_url;

   // verifica se URL é válida
   if (!(@fopen($new_url, 'r')))
     throw new Exception('Not a valid URL.');
   // tenta acrescentar bm
   add_bm($new_url) ;
   echo 'Bookmark added.';

   // obtém os bookmarks que esse usuário criou
   if ($url_array = get_user_urls($_SESSION['valid_user']))
     display_user_urls($url_array);
 }
 catch (Exception $e)
 {
   echo $e->getMessage( );
 }
 display_user_menu( );
 do_html_footer( );
?>
```

Novamente esse script segue o padrão de validação, entrada de banco de dados e saída.

Para validar, primeiro verificamos se o usuário preencheu o formulário utilizando filled_out( ). Então, realizamos duas verificações de URL. Primeiro, utilizando strstr( ), vemos se o URL inicia com http://. Se não iniciar, adicionamos isso ao início do URL. Depois que fizemos isso, podemos verificar se o URL realmente existe. Como você poderia lembrar do Capítulo 19, podemos utilizar fopen( ) para abrir um URL iniciando com http://. Se pudermos abrir esse arquivo, assumimos que o URL é válido e chamamos a função add_bm( ) para adicioná-lo ao banco de dados.

Essa função e outras relacionadas aos bookmarks estão todas na biblioteca de funções url_fns.php. Você pode ver o código para a função add_bm( ) na Listagem 26.22.

Listagem 26.22 **Função add_bm( ) a partir de url_fns.php – Essa função adiciona novos bookmarks ao banco de dados**

```
function add_bm($new_url)
{
 // Adiciona novo bookmark ao banco de dados

 echo "Attempting to add ".htmlspecialchars($new_url).'<br />';
 $valid_user = $_SESSION['valid_user'];

 $conn = db_connect( );

 // insere o novo bookmark
 if (!$conn->query( "insert into bookmark values
                          ('$valid_user', '$new_url')"))
   throw new Exception('Bookmark could not be inserted.');

 return true;
}
```

Essa função é relativamente simples. Ela verifica se um usuário já não tem esse bookmark listado no banco de dados. (Embora seja improvável que eles insiram um bookmark duas vezes, é possível, e até provável, que eles possam atualizar a página.) Se o bookmark for novo, então ele é inserido no banco de dados.

Reexaminando add_bm.php, você pode verificar que a última coisa que isso faz é chamar get_user_urls( ) e display_user_urls( ), da mesma maneira que member.php. Prosseguiremos e examinaremos essas funções em seguida.

Exibindo bookmarks

No script member.php e na função add_bm( ), utilizamos as funções get_user_urls( ) e display_user_urls( ). Essas funções obtêm os bookmarks do usuário a partir do banco de dados e os exibe, respectivamente. A função get_user_urls( ) está na biblioteca url_fns.php e a função display_user_urls( ) está na biblioteca output_fns.php.

A função get_user_urls( ) é mostrada na Listagem 26.23.

Listagem 26.23 **Função get_user_urls( ) de url_fns.php – Essa função recupera bookmarks de um usuário a partir do banco de dados**

```
function get_user_urls($username)
{
 //extrai do banco de dados todos os URLs que esse usuário armazenou
 $conn = db_connect( );
 $result = $conn->query( "select bm_URL
                          from bookmark
                          where username = '$username'");
```

Listagem 26.23 Continuação

```
  if (!$result)
    return false;

  //cria um array dos URLs
  $url_array = array( );
  for ($count = 0; $row = $result->fetch_row( ); ++$count)
  {
    $url_array[$count] = $row[0];
  }
  return $url_array;
}
```

Examinemos essa função rapidamente. Ela aceita um nome de usuário como parâmetro e recupera os bookmarks desse usuário a partir do banco de dados. Ela retornará um array desses URLs ou `false` se os bookmarks não puderem ser recuperados.

O array de `get_user_urls( )` pode ser passado para `display_user_urls( )`. Essa é novamente uma função de saída HTML simples para imprimir os URLs do usuário em um formato interessante de tabela, então não falaremos sobre ela aqui. Consulte novamente a Figura 26.6 para ver como é a saída. A função realmente coloca os URLs em um formulário. Ao lado de cada URL está uma caixa de seleção que permite que os bookmarks sejam marcados para exclusão. Veremos isso a seguir.

Excluindo bookmarks

Quando um usuário marcar alguns bookmarks para exclusão e clicar na opção `Delete BM` no menu, o formulário contendo os URLs será enviado. Cada uma das caixas de seleção é produzida pelo seguinte código na função `display_user_urls( )`:

```
echo "<td><input type=\"checkbox\" name=\"del_me[ ]\"
              value=\"$url\"></td>";
```

O nome de cada entrada é `del_me[ ]`. Isso significa que, no script de PHP ativado por esse formulário, teremos acesso a um array chamado `$del_me` que conterá todos os bookmarks a serem excluídos.

Clicar na opção `Delete BM` ativa o script `delete_bms.php`. Esse script é mostrado na Listagem 26.24.

Listagem 26.24 **`delete_bms.php`** – Esse script exclui bookmarks do banco de dados

```
<?php
 require_once('bookmark_fns.php');
 session_start( );

 //cria nomes de variáveis abreviados
 $del_me = $HTTP_POST_VARS['del_me'];
 $valid_user = $HTTP_SESSION_VARS['valid_user'];

 do_html_header('Deleting bookmarks');
 check_valid_user( );
 if (!filled_out($HTTP_POST_VARS))
 {
   echo 'You have not chosen any bookmarks to delete.
        Please try again.';
   display_user_menu( );
   do_html_footer( );
   exit;
 }
 else
```

Listagem 26.24 **Continuação**

```
{
  if (count($del_me) >0)
  {
    foreach($del_me as $url)
    {
      if (delete_bm($valid_user, $url))
        echo 'Deleted '.htmlspecialchars($url).'.<br />';
      else
        echo 'Could not delete '.htmlspecialchars($url).'.<br />';
    }
  }
  else
    echo 'No bookmarks selected for deletion';
}
// obtêm os bookmarks que esse usuário salvou
if ($url_array = get_user_urls($valid_user))
  display_user_urls($url_array);

display_user_menu( );
do_html_footer( );
?>
```

Iniciamos esse script realizando as validações normais. Quando sabemos que o usuário selecionou alguns bookmarks para exclusão, nós os excluímos no seguinte loop:

```
foreach($del_me as $url)
{
 if (delete_bm($valid_user, $url))
   echo 'Deleted '.htmlspecialchars($url).'.<br />';
 else
   echo 'Could not delete '.htmlspecialchars($url).'.<br />';
}
```

Como você pode ver, a função `delete_bm( )` faz o trabalho real de excluir o bookmark do banco de dados. Essa função é mostrada na Listagem 26.25.

Listagem 26.25 **Função `delete_bm( )` em `url_fns.php` – Essa função exclui um bookmark da lista de um usuário**

```
function delete_bm($user, $url)
{
 // exclui um URL do banco de dados
 $conn = db_connect( );
 // exclui o bookmark
 if (!$conn->query( "delete from bookmark
                      where username='$user' and bm_url='$url'"))
   throw new Exception('Bookmark could not be deleted');
 return true;
}
```

Como você pode ver, essa também é uma função muito simples. Ela tenta excluir o bookmark de um usuário particular a partir do banco de dados. Algo que deve ser notado é que queremos remover um par nome de usuário-bookmark particular. Outros usuários talvez ainda tenham esse URL com bookmark.

Alguma saída de exemplo de execução do script de exclusão em nosso sistema é mostrada na Figura 26.10.

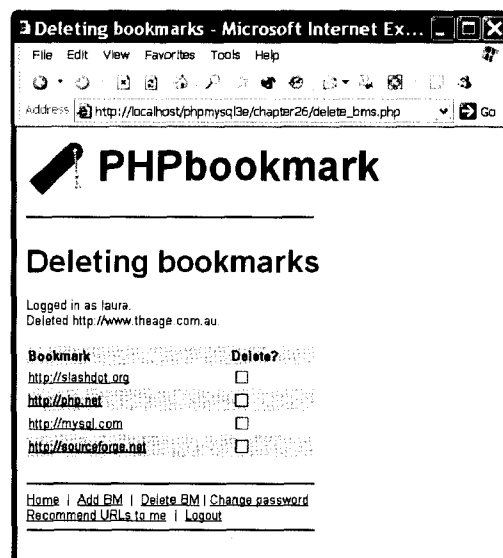



Figura 26.10 O script de exclusão notifica o usuário sobre os bookmarks excluídos e então exibe os bookmarks restantes.

Como no script `add_bms.php`, quando foram feitas as alterações no banco de dados, exibimos a nova lista de bookmark utilizando `get_user_urls()` e `display_user_urls( )`.

Implementando recomendações

Por fim, chegamos ao script de recomendação de links, `recommend.php`. Há muitas maneiras diferentes de abordar recomendações. Decidimos realizar o que chamamos de recomendação baseada em um "perfil semelhante". Isto é, procuraremos outros usuários que têm pelo menos um bookmark idêntico ao nosso usuário determinado. Os outros bookmarks daqueles outros usuários também poderiam atrair nosso dado usuário.

A maneira mais fácil de implementar isso como uma consulta SQL é utilizar subconsultas. A primeira subconsulta se parece com isto:

```
select distinct(b2.username)
                from bookmark b1, bookmark b2
                where b1.username='$valid_user'
                and b1.username != b2.username
                and b1.bm_URL = b2.bm_URL
```

Essa consulta utiliza aliases para unir a tabela de banco de dados `bookmark` a si mesma – um conceito estranho, porém útil, às vezes. Imagine que você, na verdade, possui duas tabelas de bookmark, uma chamada `b1` e a outra, `b2`. Em `b1`, vemos o usuário atual e seus bookmarks. Na outra tabela, podemos ver os bookmarks de todos os outros usuários. Estamos procurando outros usuários (`b2.username`) com a URL igual ao usuário atual (`b1.bm_URL = b2.bm_URL`) e que não seja ele (`b1.username != b2.username`).

Essa consulta fornece uma lista de pessoas com perfil semelhante ao usuário atual. De posse dessa lista, você pode procurar por outros bookmarks com a consulta externa:

```
select bm_URL
from bookmark
where username in
               (select distinct(b2.username)
                from bookmark b1, bookmark b2
               where b1.username='$valid_user'
                and b1.username != b2.username
                and b1.bm_URL = b2.bm_URL)
```

Acrescente uma segunda subconsulta para filtrar os bookmarks do usuário atual; se o usuário já possui um bookmark, não há por que recomendá-lo a ele. Finalmente, acrescente filtragem com a variável `$popularity`. Não seria bom recomendar URLs que sejam muito pessoais; portanto, sugira apenas URLs que um certo número de usuários da lista com perfil semelhante marcou com bookmark. A consulta final se parece com:

```
select bm_URL
from bookmark
where username in
                (select distinct(b2.username)
                 from bookmark b1, bookmark b2
                where b1.username='$valid_user'
                 and b1.username != b2.username
                 and b1.bm_URL = b2.bm_URL)
and bm_URL not in
                (select bm_URL
                 from bookmark
                 where username='$valid_user')
group by bm_url
having count(bm_url)>$popularity
```

Se você estivesse antecipando muitos usuários que utilizam o seu sistema, poderia ajustar `$popularity` para cima a fim de sugerir apenas URLs que foram marcados com bookmark por diversos usuários. Os URLs marcados por muitas pessoas podem ter mais qualidade e certamente possuem um apelo maior do que uma página Web normal.

O script completo para fazer recomendações é mostrado nas Listagens 26.26 e 26.27. O script principal para fazer recomendações é chamado de `recommend.php` (ver Listagem 26.26). Ele chama a função `recommend_urls( )` a partir de `url_fns.php` (ver Listagem 26.27).

Listagem 26.26 **`recommend.php` – Este Script sugere alguns bookmarks de que o usuário pode gostar**

```
<?php
 require_once('bookmark_fns.php');
 session_start( );
 do_html_header('Recommending URLs');
 try
 {
   check_valid_user( );
   $urls = recommend_urls($_SESSION['valid_user']);
   display_recommended_urls($urls);
 }
 catch(Exception $e)
 {
   echo $e->getMessage( );
 }
 display_user_menu( );
 do_html_footer( );
?>
```

Listagem 26.27 **Função `recommend_urls( )` de `url_fns.php` – Esta função elabora as recomendações reais**

```
function recommend_urls($valid_user, $popularity = 1)
{
 // Forneceremos algumas recomendações semi-inteligentes às pessoas
 // Se tiverem um URL em comum com outros usuários, eles podem gostar
 // de outros URLs de que esses outros usuários gostam
 $conn = db_connect( );
```

Listagem 26.27 **Continuação**

```
  // encontra outros usuários correspondentes
  // com um url igual ao seu
  // como uma maneira simples de excluir páginas privadas e
  // aumentar a chance de recomendar URLs interessantes,
  // especificamos um nível mínimo de popularidade
  // se $popularity = 1, então mais de uma pessoa deve ter
  // um URL antes de o recomendarmos
  $query = "select bm_URL
          from bookmark
          where username in
               (select distinct(b2.username)
                 from bookmark b1, bookmark b2
                where b1.username='$valid_user'
                 and b1.username != b2.username
                 and b1.bm_URL = b2.bm_URL)
          and bm_URL not in
                              (select bm_URL
                              from bookmark
                              where username='$valid_user')
           group by bm_url
           having count(bm_url)>$popularity";

  if (!($result = $conn->query($query)))
     throw new Exception('Could not find any bookmarks to recommend.');
  if ($result->num_rows==0)
     throw new Exception('Could not find any bookmarks to recommend.');

  $urls = array( );
  // constrói um array dos urls relevantes
  for ($count=0; $row = $result->fetch_object( ); $count++)
  {
     $urls[$count] = $row->bm_URL;
  }

  return $urls;
}
```

Um exemplo da saída de recommend.php é mostrado na Figura 26.11.

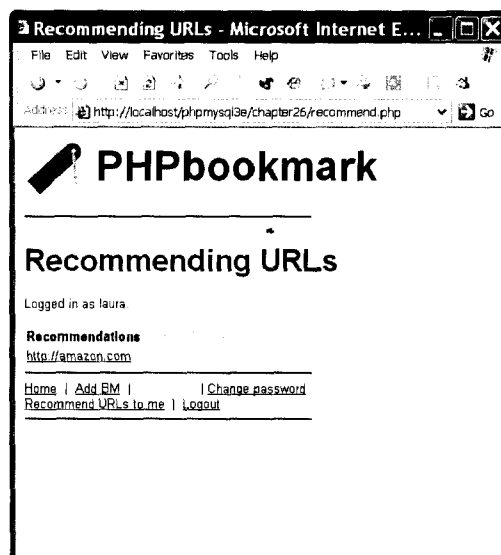



Figura 26.11 O script recommend.php recomendou que esse usuário poderia gostar da amazon.com. Pelo menos dois outros usuários no banco de dados que gostam de amazon.com também têm esse bookmark.

Empacotando e considerando possíveis extensões

Nas seções anteriores, descrevemos a funcionalidade básica da aplicação PHPbookmark. Há muitas extensões possíveis. Por exemplo, talvez você considere adicionar:

- Um agrupamento de bookmarks por tópico;
- Um link "Adicionar isso a meus bookmarks" para recomendações;
- Recomendações baseadas nos URLs mais populares no banco de dados ou sobre um tópico particular;
- Uma interface administrativa para configurar e administrar usuários e tópicos;
- Maneiras de tornar bookmarks recomendados mais inteligentes ou mais rápidos;
- Verificação adicional de erros de entrada de usuário.

Experimente! É a melhor maneira de aprender.

A seguir

No próximo projeto, construiremos um carrinho de compras que permitirá que os usuários naveguem por nosso site, adicionem compras à medida que avançam, antes de finalmente passarem no caixa e realizarem um pagamento eletrônico.

27

Construindo um carrinho de compras

NESTE CAPÍTULO, VOCÊ APRENDERÁ A CONSTRUIR um carrinho de compras básico. Adicionaremos esse carrinho de compras ao banco de dados Book-O-Rama que implementamos na Parte II. Exploraremos também outra opção – configurar e utilizar um carrinho de compras com código-fonte aberto existente em PHP.

No caso de você ainda não ter ouvido falar disso antes, o termo *carrinho de compras* (*shopping cart* – às vezes também denominado *shopping basket)* é utilizado para descrever um mecanismo específico de compras on-line. À medida que navega por um catálogo on-line, você pode adicionar itens ao seu carrinho de compras. Ao terminar de navegar, você faz o check-out, isto é, passa no caixa da loja on-line e paga pelos itens no carrinho.

Para implementar o carrinho de compras, implementaremos a seguinte funcionalidade:

- Um banco de dados dos produtos que desejamos vender on-line;
- Um catálogo de produtos on-line, listados por categoria;
- Um carrinho de compras para monitorar os itens que um usuário quer comprar;
- Um script de check-out que processa as informações sobre pagamento e endereço de entrega;
- Uma interface de administração.

O problema

Você provavelmente se lembrará do banco de dados Book-O-Rama que desenvolvemos na Parte II. Nesse projeto, instalamos e fizemos funcionar a loja on-line Book-O-Rama. Eis os requisitos desse sistema:

- Precisaremos encontrar uma maneira de conectar o banco de dados ao navegador de um usuário. Os usuários devem ser capazes de navegar pelos itens por categoria.
- Os usuários também devem poder selecionar itens do catálogo para só mais tarde concluir a compra. Precisaremos ser capazes de monitorar que itens eles selecionaram.
- Quando eles terminarem as compras, precisaremos ser capazes de somar os pedidos, obter endereços de entrega e processar o pagamento.
- Devemos também construir uma interface de administrador para o site Book-O-Rama para que o administrador possa adicionar e editar livros e categorias no site.

Componentes da solução

Vejamos as soluções para atender a cada um dos requisitos listados anteriormente.

Construindo um catálogo on-line

Já temos um banco de dados para o catálogo Book-O-Rama. Mas, provavelmente, ele precisará de algumas alterações e adições nessa aplicação. Uma delas será adicionar categorias de livros, como declarado nos requisitos.

Precisaremos também adicionar algumas informações ao nosso banco de dados existente sobre endereços de entrega, detalhes de pagamento e assim por diante. Já sabemos como construir uma interface para um banco de dados do MySQL utilizando o PHP, então essa parte da solução deve ser muito fácil.

Também devemos usar transações enquanto concluímos os pedidos dos clientes. Para isso, precisamos converter as tabelas Book-O-Rama para usar o mecanismo de armazenamento InnonDB. Esse processo também é razoavelmente direto.

Monitorando as compras de um usuário enquanto ele compra

Há duas maneiras básicas de monitorar as compras de um usuário enquanto ele compra. Uma é colocar suas seleções em nosso banco de dados e a outra é utilizar uma variável de sessão.

Utilizar uma variável de sessão para monitorar seleções de uma página para outra será mais fácil de escrever já que não exigirá que consultemos constantemente o banco de dados em busca dessas informações. Isso também evitará a situação em que acabamos com uma grande quantidade de lixo no banco de dados proveniente de usuários que estão simplesmente navegando e mudam de idéia.

Precisamos, portanto, projetar uma variável de sessão ou conjunto de variáveis para armazenar as seleções de um usuário. Quando um usuário terminar de fazer as compras a realizar o pagamento, colocaremos essas informações em nosso banco de dados como um registro de transação.

Além disso, podemos utilizar esses dados para fornecer um resumo do estado atual do carrinho em um canto da página, de modo que um usuário saiba a qualquer dado momento quanto planeja gastar.

Implementando um sistema de pagamento

Neste projeto, somaremos o pedido do usuário e anotaremos o endereço de entrega. Na realidade, não processaremos os pagamentos. Uma infinidade de sistemas de pagamento está disponível e a implementação para cada um é diferente. Escreveremos uma *função fictícia* que pode ser substituída por uma interface para seu sistema escolhido.

Geralmente os sistemas de pagamento são vendidos em áreas geográficas mais específicas que este livro. A maneira como diferentes interfaces de processamento em tempo real funcionam é genericamente semelhante. Você precisará estabelecer uma conta de comerciante junto a um banco para os cartões que você deve aceitar. Seu provedor de sistema de pagamentos especificará os parâmetros que você precisará passar para o sistema.

O sistema de pagamento transmitirá seus dados para um banco e retornará um código de sucesso ou um dos muitos diferentes tipos de códigos de erro. Em troca dos seus dados, o portal de pagamentos cobrará uma taxa de configuração ou uma taxa anual mais uma taxa baseada no número ou valor de suas transações. Alguns provedores também cobram por transações canceladas.

O sistema de pagamento escolhido precisará de informações sobre o cliente (como um número de cartão de crédito), identificar informações provenientes de você (para especificar a conta de comerciante a que o pagamento deve ser creditado) e o valor total da transação.

Podemos elaborar o total de um pedido a partir da variável de sessão do carrinho de compras de um usuário. Registraremos as informações finais do pedido no banco de dados e eliminaremos a variável de sessão nesse momento.

Construindo uma interface de administração

Além disso tudo, construiremos uma interface de administração que permitirá adicionar, excluir e editar livros e categorias a partir do banco de dados.

Uma edição comum que talvez façamos é alterar o preço de um item (por exemplo, para uma venda ou oferta especial). Isso significa que quando armazenamos o pedido de um cliente, também devemos armazenar o preço que ele paga por um item. Seria um pesadelo para a contabilidade se os únicos registros que tivéssemos fossem os itens que cada cliente encomendou e o preço atual de cada um. Isso também significa que se o cliente tiver de retornar ou trocar o item, restituiremos a quantidade correta de crédito.

Não construiremos uma interface de preenchimento e de monitoramento de pedido para esse exemplo. Você pode adicionar uma a esse sistema básico para atender às suas necessidades.

Visão geral da solução

Vamos juntar todas as partes. Há duas visualizações básicas do sistema: a visualização do usuário e a visualização do administrador. Depois de considerar a funcionalidade requerida, providenciamos dois projetos de fluxo de sistema, um para cada visualização. Esses são mostrados nas Figuras 27.1 e 27.2, respectivamente.

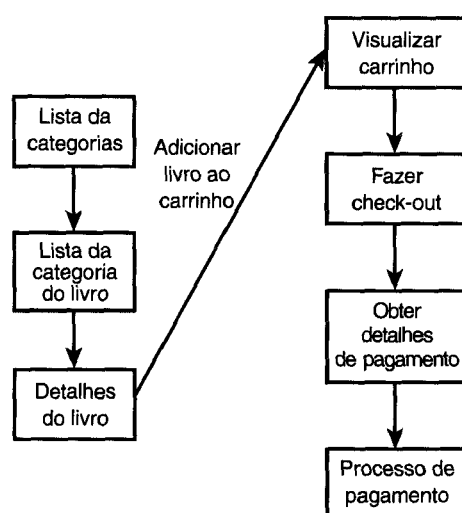



Figura 27.1 A visualização do usuário do sistema Book-O-Rama permite aos usuários navegar pelos livros por categoria, visualizar informações sobre o livro, adicionar livros ao carrinho e comprá-los.

Na Figura 27.1, mostramos os principais links entre os scripts na parte do usuário do site. Um cliente virá primeiro para a página principal, que lista todas as categorias de livros no site. A partir daí, o cliente pode chegar a uma categoria particular de livros e depois chegar às informações sobre um livro individual.

Forneceremos um link para o usuário adicionar um livro particular a seu carrinho. A partir do carrinho, ele será capaz de fazer o check-out na loja on-line.

A Figura 27.2 mostra a interface do administrador – que tem mais scripts, mas não tem muito código novo. Esses scripts permitem que um administrador efetue logon e insira livros e categorias.

A maneira mais fácil de implementar edição e exclusão de livros e categorias é mostrar ao administrador uma versão ligeiramente diferente da interface com o usuário para o site. O administrador ainda será capaz de navegar por categorias e livros, mas em vez de ter acesso ao carrinho de compras, ele poderá ir para um livro particular ou categoria e editar ou excluir esse livro ou categoria. Fazendo os mesmos scripts atenderem tanto aos usuários normais como aos usuários administradores, podemos poupar tempo e esforço.

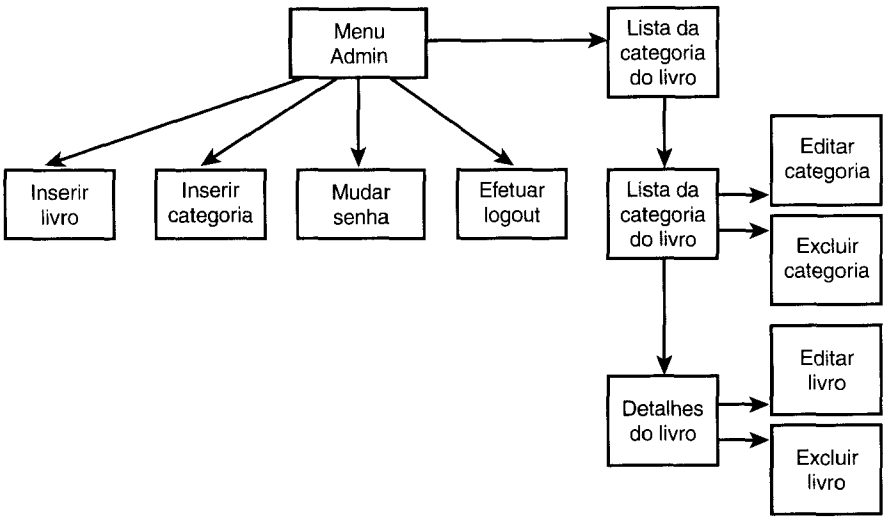



Figura 27.2 A visualização de administrador do sistema Book-O-Rama permite inserção, edição e exclusão de livros e categorias.

Os três módulos principais de código para essa aplicação são:

- Catálogo;
- Carrinho de compras e processamento de pedido (colocamos esses juntos porque estão fortemente relacionados);
- Administração.

Como no último projeto, também construiremos e utilizaremos um conjunto de bibliotecas de funções. Para esse projeto, utilizaremos uma API de função semelhante à do último projeto. Tentaremos confinar as partes de nosso código que fornecerão saída HTML a uma única biblioteca para suportar o princípio da separação de lógica e conteúdo e, sobretudo, para tornar nosso código mais fácil de ler e manter.

Além disso, precisaremos fazer algumas alterações menores no banco de dados Book-O-Rama deste projeto. Renomeamos o banco de dados `book_sc` (carrinho de compras) para distinguir o banco de dados do carrinho de compras do banco de dados construído na Parte II.

Todo o código para esse projeto pode ser encontrado no CD-ROM. Um resumo dos arquivos na aplicação é mostrado na Tabela 27.1.

Tabela 27.1 **Arquivos na aplicação carrinho de compras**

Nome	Módulo	Descrição
`index.php`	Catálogo	Página inicial principal do site para usuários. Mostra ao usuário uma lista de categorias no sistema.
`show_cat.php`	Catálogo	Mostra ao usuário todos os livros de uma categoria particular.
`show_book.php`	Catálogo	Mostra aos usuários os detalhes de um livro particular.
`show_cart.php`	Carrinho de compras	Mostra ao usuário o conteúdo do carrinho de compras. Também utilizado para adicionar itens ao carrinho.
`checkout.php`	Carrinho de compras	Apresenta ao usuário informações completas do pedido. Obtém endereço de entrega.

Tabela 27.1 **Continuação**

Nome	Módulo	Descrição
purchase.php	Carrinho de compras	Obtém detalhes de pagamento do usuário.
process.php	Carrinho de compras	Processa detalhes de pagamento e adiciona pedido ao banco de dados.
login.php	Administração	Permite ao administrador efetuar logon para fazer alterações.
logout.php	Administração	Efetua logout do usuário admin.
admin.php	Administração	Menu principal de administração.
change_password_form.php	Administração	Formulário para permitir ao administrador alterar sua senha de log.
change_password.php	Administração	Altera a senha de administrador.
insert_category_form.php	Administração	Formulário para permitir ao administrador adicionar uma nova categoria ao banco de dados.
insert_category.php	Administração	Insere nova categoria ao banco de dados.
insert_book_form.php	Administração	Formulário para permitir ao administrador adicionar um novo livro ao sistema.
insert_book.php	Administração	Insere novo livro ao banco de dados.
edit_category_form.php	Administração	Formulário para permitir ao administrador editar uma categoria.
edit_category.php	Administração	Atualiza categoria em banco de dados.
edit_book_form.php	Administração	Formulário para permitir ao administrador editar detalhes de um livro.
edit_book.php	Administração	Atualiza livro no banco de dados.
delete_category.php	Administração	Exclui uma categoria do banco de dados.
delete_book.php	Administração	Exclui um livro do banco de dados.
book_sc_fns.php	Funções	Coleção de arquivos include para essa aplicação.
admin_fns.php	Funções	Coleção de funções utilizadas por scripts administrativos.
book_fns.php	Funções	Coleção de funções para armazenar e recuperar dados de livro.
order_fns.php	Funções	Coleção de funções para armazenar e recuperar dados de pedido.
output_fns.php	Funções	Coleção de funções para dar saída HTML.
data_valid_fns.php	Funções	Coleção de funções para validar os dados de entrada.
db_fns.php	Funções	Coleção de funções para conectar-se ao banco de dados book_sc.
user_auth_fns.php	Funções	Coleção de funções para autenticar usuários administrativos.
book_sc.sql	SQL	A SQL para configurar o banco de dados book_sc.
populate.sql	SQL	A SQL para inserir alguns dados de exemplo no banco de dados book_sc.

Agora, vejamos a implementação de cada um dos módulos.

Nota Há uma grande quantidade de código nessa aplicação. Grande parte implementa funcionalidade que já vimos (particularmente no último capítulo), como armazenar dados e recuperá-los a partir do banco de dados e autenticar o usuário administrativo. Examinaremos muito rapidamente esse código, mas passaremos a maior parte do tempo nas funções de carrinho de compras.

Para o código desse projeto funcionar como escrito, você precisará ativar aspas mágicas. Se não fizer isso, você precisará adicionar com `addslashes( )` aos dados que irão para o banco de dados do MySQL, e tirar com `stripslashes( )` dos dados voltando do banco de dados. Utilizamos isso como um atalho útil.

Você pode ativar aspas mágicas individualmente por diretório em um arquivo `.htaccess` com a diretiva:

```
php_value magic_quotes_gpc on
```

Implementando o banco de dados

Como mencionamos anteriormente, fizemos algumas modificações menores ao banco de dados Book-O-Rama apresentado na Parte II.

A SQL para criar o banco de dados `book_sc` é mostrada na Listagem 27.1.

Listagem 27.1 **book_sc.sql** – SQL para criar o banco de dados **book_sc**

```
create database book_sc;

use book_sc;

create table customers
(
  customerid int unsigned not null auto_increment primary key,
  name char(40) not null,
  address char(40) not null,
  city char(20) not null,
  state char(20),
  zip char(10),
  country char(20) not null
)type=InnoDB;

create table orders
(
  orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null references customers(customerid),
  amount float(6,2),
  date date not null,
  order_status char(10),
  ship_name char(40) not null,
  ship_address char(40) not null,
  ship_city char(20) not null,
  ship_state char(20),
  ship_zip char(10),
  ship_country char(20) not null
) type=InnoDB;;

create table books
(
   isbn char(13) not null primary key,
   author char(30),
  title char(60),
   catid int unsigned,
   price float(4,2) not null,
   description varchar(255)
```

Listagem 27.1 Continuação

```
) type=InnoDB;;

create table categories
(
  catid int unsigned not null auto_increment primary key,
  catname char(40) not null
) type=InnoDB;;

create table order_items
(
  orderid int unsigned not null references orders(orderid),
  isbn char(13) not null,
  item_price float(4,2) not null,
  quantity tinyint unsigned not null,
  primary key (orderid, isbn)
) type=InnoDB;;

create table admin
(
  username char(16) not null primary key,
  password char(16) not null
);

grant select, insert, update, delete
on book_sc.*
to book_sc@localhost identified by 'password';
```

Embora nada estivesse errado com a interface Book-O-Rama, temos alguns outros requisitos que agora tornaremos disponíveis on-line.

As alterações que fizemos no banco de dados original são:

- A adição de mais campos de endereço para clientes – o mais importante agora é que estamos construindo uma aplicação mais realista.
- A adição do endereço de entrega a um pedido. O endereço para entrar em contato com o cliente pode não ser o mesmo que o endereço de entrega, especialmente se ele estiver utilizando o site para comprar um presente.
- A adição de uma tabela `categorias` e uma tabela `catid to books`. Classificar livros em categorias facilitará mais a navegação.
- A adição de `item_price` à tabela `order_items` para reconhecer o fato de que o preço de um item pode mudar. Queremos saber quanto custa quando o cliente fizer o pedido.
- A adição de uma tabela `admin` para armazenar detalhes de login de administrador e senha.
- A remoção da tabela de revisões. Você poderia adicionar revisões como uma extensão para esse projeto. Em vez disso, cada livro tem um campo de descrição que conterá uma breve publicidade do livro.
- A mudança de mecanismos de armazenamento para InnoDB. Faça isso para que você possa utilizar chaves estrangeiras e também para que possa usar transações ao inserir informações de pedidos de clientes.

Para configurar esse banco de dados no sistema, execute o `script book_sc.sql` por meio do MySQL como o usuário root, da seguinte maneira:

```
mysql -u root -p < book_sc.sql
```

(Você precisará fornecer sua senha root.)

De antemão, você deve alterar a senha para o usuário book_sc para algo melhor que 'password'. Observe que se alterar a senha em book_sc.sql, você também precisará alterá-la em db_fns.php. (Veremos onde em breve.)

Incluímos também um arquivo de dados de exemplo. Ele se chama populate.sql. Você pode colocar os dados de exemplo no banco de dados executando-o pelo MySQL dessa mesma maneira.

Implementando o catálogo on-line

Três scripts de catálogo estão nessa aplicação: a página principal, a página de categorias e a página de detalhes do livro.

A página frontal do site é produzida pelo script chamado index.php. A saída desse script é mostrada na Figura 27.3.

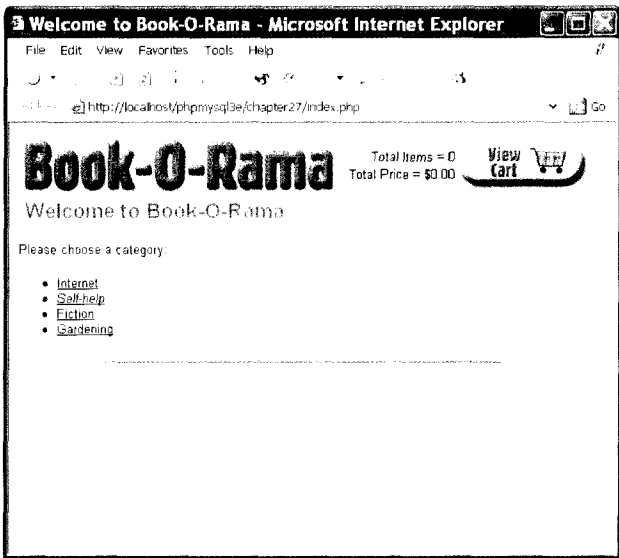



Figura 27.3 A página inicial do site lista as categorias de livros disponíveis para compra.

Você notará que, além da lista de categorias no site, há um link para o carrinho de compras no canto superior direito da tela e algumas informações de resumo sobre o que há no carrinho. Isso aparecerá em todas as páginas enquanto um usuário navega e compra.

Se um usuário clicar em uma das categorias, ele será levado à página de categorias, produzida pelo script show_cat.php. A página de categorias para a seção de livros da Internet é mostrada na Figura 27.4.

Todo os livros na categoria Internet são listados como links. Se um usuário clicar em um desses links, ele será levado para a página de detalhes do livro. A página de detalhes do livro é mostrada na Figura 27.5.

Nessa página, bem como no link View Cart, temos um link Add to Cart em que o usuário pode selecionar um item. Voltaremos para esse ponto quando examinarmos a maneira como construir o carrinho de compras mais adiante.

Vejamos cada um desses três scripts.

Listando categorias

O primeiro script usado neste projeto, index.php, lista todas as categorias no banco de dados. Esse código é mostrado na Listagem 27.2.

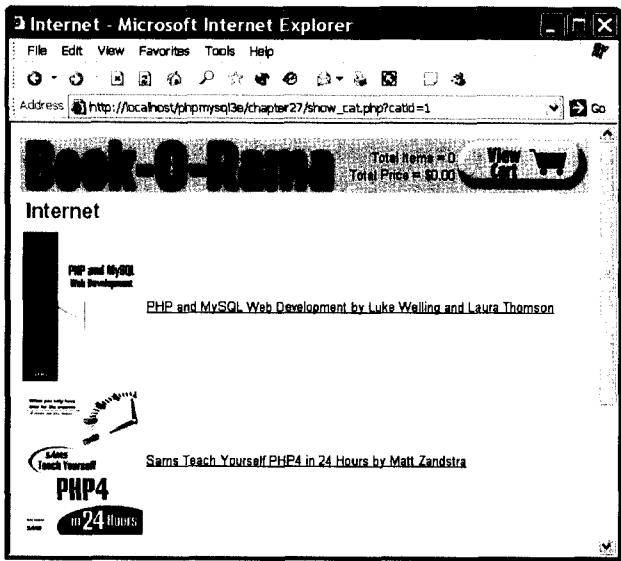



Figura 27.4 Cada livro na categoria é listado com uma foto.

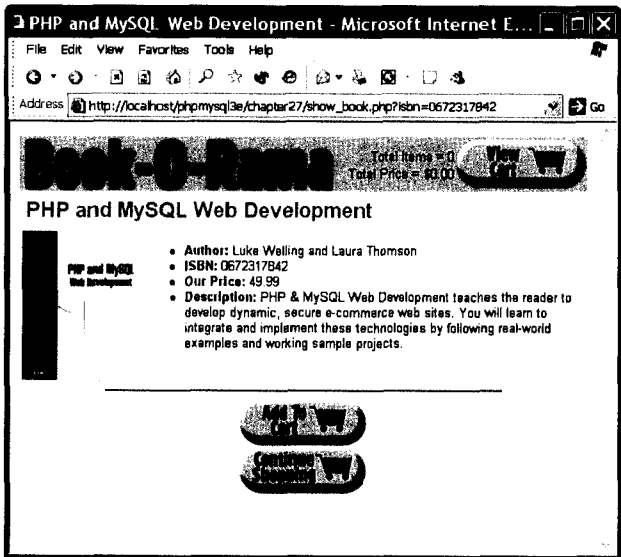



Figura 27.5 Cada livro tem uma página de detalhes que mostra informações adicionais incluindo uma descrição longa.

Listagem 27.2 **index.php – Script para produzir a página inicial do site**

```
<?php
  include ('book_sc_fns.php');
  // O carrinho de compras precisa de sessões, então inicia uma
  session_start( );
  do_html_header('Welcome to Book-O-Rama');

  echo '<p>Please choose a category:</p>';

  // obtém as categorias do banco de dados
  $cat_array = get_categories( );

  // exibe como links para páginas de categoria
  display_categories($cat_array);
```

Listagem 27.2 **Continuação**

```
  // se efetuou logon como admin, mostra links de adicionar, excluir e editar cat
  if(isset($_SESSION['admin_user']))
  {
    display_button('admin.php', 'admin-menu', 'Admin Menu');
  }
  do_html_footer( );
?>
```

O script começa incluindo `book_sc_fns.php`, o arquivo que inclui todas as bibliotecas de funções para essa aplicação.

Depois disso, devemos iniciar uma sessão. Isso será necessário para que o carrinho de compras funcione. Cada página no site utilizará a sessão.

Há algumas chamadas para funções de saída de HTML como `do_html_header( )` e `do_html_footer( )` (as duas contidas em `output_fns.php`).

Também temos algum código que verifica se o usuário está conectado como um administrador e fornece algumas opções diferentes de navegação se ele estiver – retornaremos a isso na seção sobre funções de administração.

A parte mais importante desse script é:

```
// obtém categorias do banco de dados
$cat_array = get_categories( );

// exibe como links para páginas de categoria
display_categories($cat_array);
```

As funções `get_categories( )` e `display_categories( )` estão nas bibliotecas de funções `book_fns.php` e `output_fns.php`, respectivamente. A função `get_categories( )` retorna um array das categorias no sistema, que então passamos para `display_categories( )`. Vejamos o código para `get_categories( )`. Esse código é mostrado na Listagem 27.3.

Listagem 27.3 **Função `get_categories( )` de `book_fns.php` – Função que recupera uma lista de categoria do banco de dados**

```
function get_categories( )
{
  // consulta uma lista de categorias no banco de dados
   $conn = db_connect( );
   $query = 'select catid, catname
             from categories';
   $result = @$conn>_query($query);
   if (!$result)
     return false;
   $num_cats = @mysql_num_rows($result);
   if ($num_cats ==0)
      return false;
   $result = db_result_to_array($result);
   return $result;
}
```

Como você pode ver, essa função conecta-se ao banco de dados e recupera uma lista de todos os IDs de categoria e nomes. Escrevemos e utilizamos uma função chamada `db_result_to_array( )`, encontrada em `db_fns.php`. Essa função é mostrada na Listagem 27.4. Ela aceita um identificador de resultado do MySQL e retorna um array numericamente indexado de linhas, onde cada linha é um array associativo.

Listagem 27.4 **Função `db_result_to_array( )` de `db_fns.php` – Função que converte um identificador de resultados do MySQL em um array de resultados**

```
function db_result_to_array($result)
{
   $res_array = array( );

   for ($count=0; $row = $result->fetch_assoc( ); $count++)
     $res_array[$count] = $row;

   return $res_array;
}
```

Nesse caso, retornaremos esse array de volta a index.php, onde o passamos para a função `display_categories( )` de `output_fns.php`. Essa função exibe cada categoria como um link para a página que contém os livros nessa categoria. O código para essa função é mostrado na Listagem 27.5.

Listagem 27.5 **Função `display_categories( )` de `output_fns.php` – Função que exibe um array de categorias como uma lista de links para essas categorias**

```
function display_categories($cat_array)
{
  if (!is_array($cat_array))
  {
     echo 'No categories currently available<br />';
     return;
  }
  echo '<ul>';
  foreach ($cat_array as $row)
  {
    $url = 'show_cat.php?catid='.($row['catid']);
    $title = $row['catname'];
    echo '<li>';
    do_html_url($url, $title);
    echo '</li>
  }
  echo '</ul>';
  echo '<hr />';
}
```

Essa função converte cada categoria do banco de dados em um link. Cada link vai para o próximo script – `show_cat.php` – mas cada um tem um parâmetro diferente, o ID de categoria ou `catid`. (Esse é um número único, gerado pelo MySQL e utilizado para identificar a categoria.)

Esse parâmetro para o próximo script determinará qual categoria acabaremos vendo.

Listando livros em uma categoria

O processo para listar livros em uma categoria é semelhante. O script que faz isso se chama `show_cat.php`. Esse código é mostrado na Listagem 27.6.

Listagem 27.6 **`show_cat.php` – Esse script mostra os livros em uma categoria particular**

```
<?php
 require ('book_sc_fns.php');
 // O carrinho de compras precisa de sessões, então comece uma
 session_start( );
```

Listagem 27.6 **Continuação**

```
$catid = $_GET['catid'];
$name = get_category_name($catid);

do_html_header($name);

// obtém informações do livro a partir do db
$book_array = get_books($catid);

display_books($book_array);


// se conectado como admin, mostra, adiciona, exclui links de livros
if(isset($_SESSION['admin_user']))
{
  display_button('index.php', 'continue', 'Continue Shopping');
  display_button('admin.php', 'admin-menu', 'Admin Menu');
  display_button("edit_category_form.php?catid=$catid",
   'edit-category', 'Edit Category');
}
else
  display_button('index.php', 'continue-shopping', 'Continue Shopping');

do_html_footer( );
?>
```

Esse script é muito semelhante em estrutura à página de índice, a diferença é que estamos recuperando livros em vez de categorias.

Iniciamos com `session_start( )` como de costume e então convertemos o ID de categoria que nos foi passado em um nome de categoria utilizando a função `get_category_name( )` assim:

```
$name = get_category_name($catid);
```

Essa função, mostrada na Listagem 27.7, pesquisa o nome de categoria no banco de dados.

Listagem 27.7 **Função `get_category_name( )` de `book_fns.php` – Essa função converte um ID de categoria em um nome de categoria**

```
function get_category_name($catid)
{
  // consulta o nome de um id de categoria no banco de dados
   $conn = db_connect( );
   $query = "select catname
             from categories
             where catid = $catid";
   $result = @mysql_query($query);
   if (!$result)
     return false;
   $num_cats = @$result_num_rows;
   if ($num_cats ==0)
      return false;
   $row = $result ->fetch_object( );
   return $row->catname;
}
```

Depois que recuperamos o nome de categoria, podemos exibir um cabeçalho HTML e prosseguir para recuperar e listar os livros do banco de dados que caem na nossa categoria escolhida, da seguinte maneira:

```
$book_array = get_books($catid);
display_books($book_array);
```

As funções get_books( ) e display_books( ) são extremamente semelhantes às funções get_categories( ) e display_categories( ); portanto, não serão abordadas aqui. A única diferença é que recuperamos informações a partir da tabela de livros em vez da tabela de categorias.

A função display_books( ) fornece um link para cada livro na categoria por meio do script show_book.php. Novamente, cada link é sufixado com um parâmetro. Mais ou menos nesse momento, entra em questão o ISBN do livro.

Na parte inferior do script show_cat.php, você verá que há algum código para exibir algumas funções adicionais se um administrador estiver conectado. Veremos essas funções na seção sobre funções administrativas.

Mostrando detalhes do livro

O script show_book.php aceita um ISBN como um parâmetro e recupera e exibe os detalhes desse livro. O código para esse script é mostrado na Listagem 27.8.

Listagem 27.8 **show_book.php – Esse script mostra os detalhes de um livro particular**

```
<?php
require ('book_sc_fns.php');
// O carrinho de compras precisa de sessões, então comece uma
session_start( );

$isbn = $_GET['isbn'];

// tira esse livro do banco de dados
$book = get_book_details($isbn);
do_html_header($book['title']);
display_book_details($book);

// configura url para o botão "continue"
$target = 'index.php';
if($book['catid'])
{
  $target = 'show_cat.php?catid='.$book['catid'];
}
// se conectado como admin, mostra links de edição de livro
if( check_admin_user( ) )
{
  display_button("edit_book_form.php?isbn=$isbn", 'edit-item', 'Edit Item');
  display_button('admin.php', 'admin-menu', 'Admin Menu');
  display_button($target, 'continue', 'Continue');
}
else
{
  display_button("show_cart.php?new=$isbn", 'add-to-cart', 'Add '
               .$book['title'].' To My Shopping Cart');
  display_button($target, 'continue-shopping', 'Continue Shopping');
}

do_html_footer( );
?>
```

Novamente com esse script, estamos fazendo coisas muito semelhantes às das duas páginas anteriores. Começamos iniciando a sessão e então utilizando

```
$book = get_book_details($isbn);
```

para obter as informações de livro do banco de dados, e

```
display_book_details($book);
```

para enviar para saída os dados em HTML.

Algo a ser notado é que `display_book_details( )` procura um arquivo de imagem para o livro como `images/$isbn.jpg`. Se esse arquivo não existir, nenhuma imagem será exibida.

O restante do script configura a navegação. Um usuário normal terá as escolhas Continue Shopping, que o levará de volta à página de categorias, e Add to Cart, que adicionará o livro ao seu carrinho de compras. Se um usuário estiver conectado como um administrador, ele obterá algumas opções diferentes, que veremos na seção sobre administração.

Isso completa os princípios básicos do sistema de catálogo. Vamos seguir adiante e ver o código para a funcionalidade do carrinho de compras.

Implementando o carrinho de compras

A funcionalidade do carrinho de compras gira inteiramente em torno de uma variável de sessão chamada `cart`. Esse é um array associativo que tem ISBNs como chaves e quantidades como valores. Por exemplo, se eu adicionasse uma única cópia desse livro ao meu carrinho de compras, o array conteria:

```
0672317842 => 1
```

Isto é, uma cópia do livro com o ISBN 0672317842. Quando adicionarmos itens ao carrinho, eles serão adicionados ao array. Quando visualizarmos o carrinho, utilizaremos o array `$cart` para pesquisar os detalhes completos dos itens no banco de dados.

Também utilizamos duas outras variáveis de sessão para controlar a exibição no cabeçalho que mostra Total Items e Total Price. Essas variáveis são chamadas `items` e `total_price`, respectivamente.

Utilizando o script `show_cart.php`

Começaremos entendendo como o código do carrinho de compras é implementado examinando o script `show_cart.php`. Esse é o script que exibe a página que visitaremos se clicarmos em qualquer link Cart ou Add to Cart. Se chamarmos `show_cart.php` sem nenhum parâmetro, conseguiremos ver o conteúdo dele. Se o chamarmos com um ISBN como parâmetro, o item com esse ISBN será adicionado ao carrinho.

Para entender isso completamente, veja primeiro a Figura 27.6.

Nesse caso, clicamos no link View Cart quando o carrinho está vazio; isto é, ainda não selecionamos nenhum item para compra.

A Figura 27.7 mostra o carrinho um pouco mais para baixo quando selecionamos dois livros para compra. Nesse caso, chegamos nessa página clicando no link Add to Cart na página show_book.php desse livro, cujo título em inglês é *PHP and MySQL Web Development*. Se examinar o URL de perto, você verá que chamamos o script com um parâmetro desta vez. O parâmetro é chamado new e tem o valor 0672317842 – isto é, o ISBN para o livro que acabamos de adicionar ao carrinho.

Dessa página, você pode ver que temos duas outras opções. Há um botão Save Changes que podemos utilizar para alterar a quantidade de itens no carrinho. Para fazer isso, você pode alterar as quantidades diretamente e clicar em Save Changes. Esse é realmente um botão de enviar que nos leva de volta para o script `show_cart.php` novamente para atualizar o carrinho.

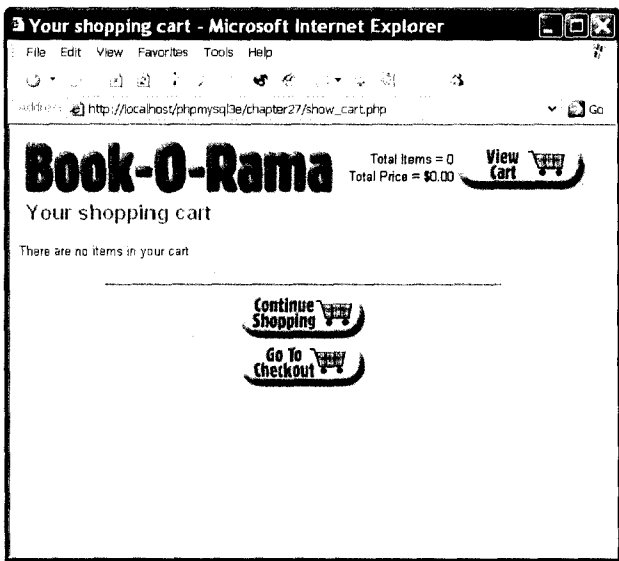



Figura 27.6 O script `show_cart.php` sem parâmetros mostra somente o conteúdo do carrinho.

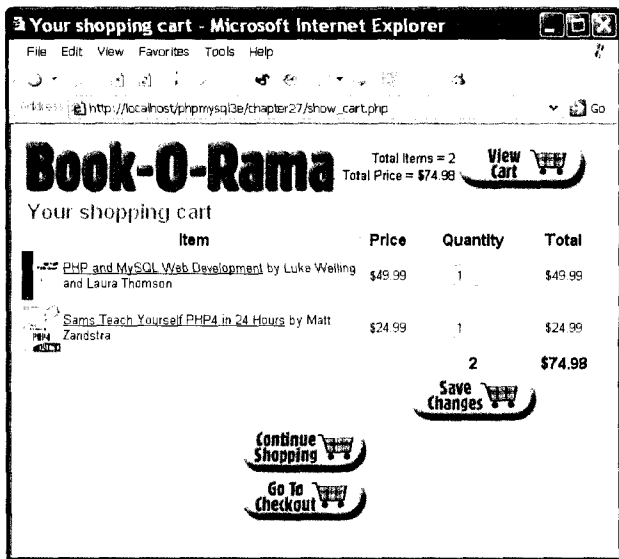



Figura 27.7 O script `show_cart.php` com o novo parâmetro adiciona um novo item ao carrinho.

Além disso, há um botão Go to Checkout em que um usuário pode clicar quando estiver pronto para partir. Voltaremos para esse em um minuto.

Por enquanto, vejamos o código para o script `show_cart.php`. Esse código é mostrado na Listagem 27.9.

Listagem 27.9 **`show_cart.php` – Esse script controla o carrinho de compras**

```
<?php
require ('book_sc_fns.php');
// O carrinho de compras precisa de sessões, então inicia uma
session_start( );

@ $new = $_GET['new'];
```

Listagem 27.9 **Continuação**

```
if($new)
{
  //novo item selecionado
  if(!isset($_SESSION['cart']))
  {
    $_SESSION['cart'] = array( );
    $_SESSION['items'] = 0;
    $_SESSION['total_price'] ='0.00';
  }

  if(isset($_SESSION['cart'][$new]))
    $_SESSION['cart'][$new]++;
  else
    $_SESSION['cart'][$new] = 1;

  $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
  $_SESSION['items'] = calculate_items($_SESSION['cart']);
}

if(isset($_POST['save']))
{
  foreach ($_SESSION['cart'] as $isbn => $qty)
  {
    if($_POST[$isbn]=='0')
      unset($_SESSION['cart'][$isbn]);
    else
      $_SESSION['cart'][$isbn] = $_POST[$isbn];
  }
  $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
  $_SESSION['items'] = calculate_items($_SESSION['cart']);
}

do_html_header('Your shopping cart') ;

if($_SESSION['cart']&&array_count_values($_SESSION['cart']))
  display_cart($_SESSION['cart']);
else
{
  echo '<p>There are no items in your cart</p>';
  echo '<hr />';
}
$target = 'index.php';

// se acabamos de adicionar um item ao carrinho
// continue comprando nesta categoria
if($new)
{
  $details =  get_book_details($new);
  if($details['catid'])
    $target = 'show_cat.php?catid='.$details['catid'];
}
display_button($target, 'continue-shopping', 'Continue Shopping');

// use isso se SSL estiver configurado
// $path = $_SERVER['PHP_SELF'];
// $server = $_SERVER['SERVER_NAME'];
// $path = str_replace('show_cat.php', '', $path);
// exibe botões('https://'.$server.$path.'checkout.php',
//                  'vá para o checkout', 'Go To Checkout');
```

Listagem 27.9 **Continuação**

```
// se não houver SSL use o código a seguir
display_button('checkout.php', 'go-to-checkout', 'Go To Checkout');


do_html_footer( );
?>
```

Há três partes principais para esse script: exibir o carrinho, adicionar itens ao carrinho e salvar alterações no carrinho. Abordaremos essas partes nas próximas três seções.

Visualizando o carrinho

Independente da página de onde viemos, exibiremos o conteúdo do carrinho. No caso básico, quando um usuário acabou de clicar em View Cart, essa é a única parte do código que será executada, da seguinte maneira:

```
if($_SESSION['cart']&&array_count_values($_SESSION['cart']))
   display_cart($_SESSION['cart']);
else
{
 echo '<p>There are no items in your cart</p>';
 echo '<hr />';
}
```

Como você pode ver a partir desse código, se tivermos um carrinho com algum conteúdo, chamaremos a função `display_cart( )`. Se o carrinho estiver vazio, forneceremos uma mensagem para o usuário para esse efeito.

A função `display_cart( )` simplesmente imprime o conteúdo do carrinho como um formato HTML legível, como você pode ver nas Figuras 27.6 e 27.7. O código para essa função pode ser encontrado em `output_fns.php`, que está incluído aqui como a Listagem 27.10. Embora ele seja uma função de exibição, esse código é razoavelmente complexo, então o incluímos aqui.

Listagem 27.10 **Função `display_cart( )` de `output_fns.php` – Essa função formata e imprime o conteúdo do carrinho de compras**

```
function display_cart($cart, $change = true, $images = 1)
{
 // exibe itens no carrinho de compras
 // opcionalmente permite alterações (true ou false)
 // opcionalmente inclui imagens (1 - sim, 0 - não)

  echo '<table border = "0" width = "100%" cellspacing = "0">
       <form action = "show_cart.php" method = "post">
       <tr><th colspan = '. (1+$images) .' bgcolor="#cccccc">Item</th>
       <th bgcolor="#cccccc">Price</th><th bgcolor="#cccccc">Quantity</th>
       <th bgcolor="#cccccc">Total</th></tr>';

 //exibe cada item como uma linha de tabela
 foreach ($cart as $isbn => $qty)
 {
   $book = get_book_details($isbn);
   echo '<tr>';
   if($images ==true)
   {
     echo '<td align = left>';
     if (file_exists("images/$isbn.jpg"))
```

Listagem 27.10 **Continuação**

```
      {
        $size = GetImageSize('images/'.$isbn.'.jpg');
        if($size[0]>0 && $size[1]>0)
        {
          echo '<img src="images/'.$isbn.'.jpg" border="0" ';
          echo 'width = '. $size[0]/3 .' height = ' .$size[1]/3 . ' />';
        }
      }
      else
        echo ' ';
      echo '</td>';
    }
    echo '<td align = "left">';
    echo '<a href = "show_book.php?isbn='.$isbn.'">'
        .$book['title'].'</a> by '.$book['author'];
    echo '</td><td align = "center">$'.number_format($book['price'], 2);
    echo '</td><td align = "center">';
    // se permitirmos alterações, quantidades estarão nas caixas de texto
    if ($change == true)
      echo "<input type = 'text' name = \"$isbn\" value = \"$qty\"
             size = \"3\" />";
    else
      echo $qty;
    echo '</td><td align = "center">$'.number_format($book['price']*$qty,2)
        ."</td></tr>\n";
  }
  // exibe linha total
  echo "<tr>
          <th colspan = ". (2+$images) ." bgcolor=\"#cccccc\"> </td>
          <th align = \"center\" bgcolor=\"#cccccc\">
              ".$_SESSION['items']."
          </th>
          <th align = \"center\" bgcolor=\"#cccccc\">
              \$".number_format($_SESSION['total_price'], 2).
          '</th>
        </tr>';
  // display save change button
  if($change == true)
  {
    echo '<tr>
            <td colspan = '. (2+$images) .'> </td>
            <td align = "center">
              <input type = "hidden" name = "save" value = "true" />
              <input type = "image" src = "images/save-changes.gif"
                     border = "0" alt = "Save Changes" />
            </td>
            <td> </td>
        </tr>';
  }
  echo '</form></table>';
}
```

O fluxo básico dessa função é:

1. Realizamos um loop por cada item no carrinho e passamos o ISBN de cada item para `get_book_details( )` a fim de que possamos resumir os detalhes de cada livro.

2. Fornecemos uma imagem para cada livro, se existir alguma. Utilizamos as tags `height` e `width` da imagem de HTML para diminuir um pouco o tamanho da imagem aqui. Isso significa que

as imagens serão um pouco distorcidas, mas elas são pequenas o bastante para isso não representar um grande problema. (Se isso incomodar, você sempre pode redimensionar as imagens utilizando a biblioteca gd discutida no Capítulo 21 ou gerar manualmente diferentes tamanhos de imagens para cada produto.)

3. Tornamos cada entrada de carrinho um link para o livro apropriado, isto é, para show_book.php com o ISBN como um parâmetro.

4. Se estivermos chamando a função com o parâmetro change configurado como true (ou não configurado – caso em que o padrão de true é assumido), mostramos as caixas com as quantidades como um formulário com o botão Save Changes no final. (Quando reutilizarmos essa função depois de fazer o check-out, não desejaremos que o usuário seja capaz de alterar o pedido.)

Nada é terrivelmente complicado nessa função, mas ela faz bastante trabalho, então talvez você ache útil ler essa função cuidadosamente.

Adicionando itens ao carrinho

Se um usuário veio para a página show_cart.php clicando em um botão Add to Cart, temos de trabalhar um pouco antes de mostrarmos a ele o conteúdo do carrinho. Especificamente, precisamos adicionar o item apropriado ao carrinho, como é mostrado a seguir.

Primeiro, se o usuário não colocou nenhum item no carrinho antes, ele não terá um carrinho, então precisamos criar um:

```
if(!isset($_SESSION['cart']))
{
  $_SESSION['cart'] = array( );
  $_SESSION['items'] = 0;
  $_SESSION['total_price'] ='0.00';
}
```

Para começar, o carrinho está vazio.

Segundo, depois que sabemos que um carrinho está configurado, podemos adicionar o item a ele:

```
if(isset($_SESSION['cart'][$new]))
  $_SESSION['cart'][$new]++;
else
  $_SESSION['cart'][$new] = 1;]
```

Aqui, estamos verificando se o item está já no carrinho. Se estiver, incrementamos a quantidade desse item no carrinho por um. Se não, adicionamos o novo item ao carrinho.

Terceiro, precisamos elaborar o preço total e o número de itens no carrinho. Para isso, utilizamos as funções calculate_price( ) e calculate_items( ), assim:

```
$_SESSION['total_price'] = calculate_price($_SESSION['cart']);
$_SESSION['items'] = calculate_items($_SESSION['cart']);
```

Essas funções são encontradas na biblioteca de funções book_fns.php. O código para eles é mostrado nas Listagens 27.11 e 27.12, respectivamente.

Listagem 27.11 **Função calculate_price( ) de book_fns.php – Essa função calcula e retorna o preço total do conteúdo do carrinho de compras**

```
function calculate_price($cart)
{
  // soma o preço total de todos os itens no carrinho de compras
  $price = 0.0;
```

Listagem 27.11 **Continuação**

```
  if(is_array($cart))
  {
    $conn = db_connect( );
    foreach($cart as $isbn => $qty)
    {
      $query = "select price from books where isbn='$isbn'";
      $result = mysql_query($query);     if ($result)
      {
        $item = $result->fetch_object( );
       $item_price = $item->price;
       $price +=$item_price*$qty;
      }
    }
  }
 return $price;
}
```

Como você pode ver, a função `calculate_price( )` trabalha pesquisando o preço de cada item no carrinho do banco de dados. Isso é um pouco lento, então evite fazer isso com mais freqüência que o necessário, armazenaremos o preço (e o número total de itens, também) como variáveis de sessão e somente recalcularemos quando o carrinho se alterar.

Listagem 27.12 **Função `calculate_items( )` de `book_fns.php` – Essa função calcula e retorna o número total de itens no carrinho de compras**

```
function calculate_items($cart)
{
  // soma o total de itens no carrinho de compras
  $items = 0;
  if(is_array($cart))
  {
    $items=array_sum($cart);
  }
 return $items;
}
```

A função `calculate_items( )` é mais simples – essa função simplesmente examina o carrinho e soma as quantidades de cada item para obter o número total de itens usando a função `array_sum( )`. Se ainda não tiver um array (se o carrinho estiver vazio), retorna 0(zero).

Salvando o carrinho atualizado

Se chegamos no `script show_cart.php` a partir do clique no botão Save Changes, o processo é um pouco diferente. Nesse caso, chegamos via envio de um formulário. Se examinar mais de perto o código, você verá que o botão Save Changes é o botão de enviar de um formulário. Esse formulário contém a variável oculta `save`. Se essa variável estiver configurada, sabemos que chegamos a esse script a partir do botão Save Changes. Isso significa que o usuário presumivelmente editou os valores de quantidade no carrinho e precisamos atualizá-los.

Se examinar outra vez as caixas de texto na parte do formulário Save Changes do script, você verá que seus nomes são derivados do ISBN do item que elas representam, assim:

```
echo '<input type="text" name="$isbn" value="$qty" size="3">';
```

Agora veja a parte do script que salva as alterações:

```
if(isset($_POST['save']))
{
 foreach ($_SESSION['cart'] as $isbn => $qty)
 {
   if($_POST[$isbn]=='0')
     unset($_SESSION['cart'][$isbn]);
   else
     $_SESSION['cart'][$isbn] = $_POST[$isbn];
 }
 $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
 $_SESSION['items'] = calculate_items($_SESSION['cart']);
}
```

Você pode ver que estamos estabelecendo nosso caminho pelo carrinho de compras, e para cada `isbn` no carrinho, estamos verificando a variável `POST` com esse nome. Esses são os campos de formulário do formulário Save Changes.

Se algum dos campos for configurado como zero, removemos totalmente esse item do carrinho de compras, utilizando `unset( )`. Caso contrário, atualizamos o carrinho para corresponder aos campos de formulário, da seguinte maneira:

```
if($_POST[$isbn]=='0')
 unset($_SESSION['cart'][$isbn]);
else
 $_SESSION['cart'][$isbn] = $_POST[$isbn];
```

Depois dessas atualizações, novamente utilizamos `calculate_price( )` e `calculate_items( )` para elaborar os novos valores das variáveis de sessão de `total_price` e `items`.

Imprimindo um resumo na barra de cabeçalho

Você já deve ter observado que na barra de cabeçalho de cada uma das páginas no site é apresentado um resumo do que está no carrinho de compras. Isso é obtido imprimindo os valores das variáveis de sessão `total_price` e `items`. Isso é feito na função `do_html_header( )`.

Essas variáveis são registradas quando o usuário visita a página `show_cart.php` pela primeira vez. Também precisamos de algumas lógicas para lidar com os casos em que um usuário ainda não visitou essa página. Essa lógica também está na função `do_html_header( )`:

```
if(!$_SESSION['items']) $_SESSION['items'] = '0';
if(!$_SESSION['total_price']) $_SESSION['total_price'] = '0.00';
```

Fazendo check-out

Quando o usuário clica no botão Go to Checkout a partir do carrinho de compras, ativa o script `checkout.php`. A página de verificação e as páginas atrás devem ser acessadas via SSL, mas a aplicação de exemplo não força a fazer isso. (Para ler mais sobre SSL, revise o Capítulo 17.)

A página de check-out é mostrada na Figura 27.8.

Esse script exige que o cliente insira seu endereço (e endereço de entrega se for diferente). Esse script é bem simples, o que você pode ver examinando o código na Listagem 27.13.

Listagem 27.13 **`checkout.php` – Esse script obtém as informações pessoais do cliente**

```
<?php
 //inclui nosso conjunto de funções
 require ('book_sc_fns.php');

 // O carrinho de compras precisa de sessões, então inicie uma
 session_start( );
```

Listagem 27.13 **Continuação**

```
do_html_header('Checkout');

if($_SESSION['cart']&&count($_SESSION['cart']))
{
  display_cart($_SESSION['cart'], false, 0);
  display_checkout_form( );
}
else
  echo '<p>There are no items in your cart</p>';

display_button('show_cart.php', 'continue-shopping', 'Continue Shopping');

do_html_footer( );
?>
```

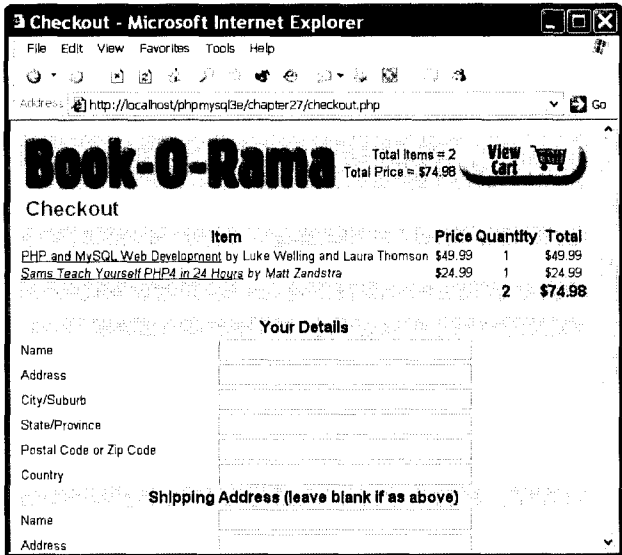



Figura 27.8 O script checkout.php obtém as informações pessoais do cliente.

Não há grandes surpresas nesse script. Se o carrinho estiver vazio, o script notificará o cliente; caso contrário, ele exibirá o formulário que você pode ver na Figura 27.8.

Se um usuário continuar clicando no botão Purchase na parte inferior do formulário, ele será levado até o script purchase.php. Você pode ver a saída desse script na Figura 27.9.

O código para esse script é ligeiramente mais complicado que o código checkout.php. Esse código é mostrado na Listagem 27.14.

Listagem 27.14 **purchase.php – Esse script armazena os detalhes do pedido no banco de dados e obtém os detalhes de pagamento**

```
<?php

 include ('book_sc_fns.php');
 // O carrinho de compras precisa de sessões, então inicie uma
 session_start( );

 do_html_header("Checkout");
 // cria os nomes de variáveis abreviados
 $name = $_POST['name'];
```

Listagem 27.14 Continuação

```
$address = $_POST['address'];
$city = $_POST['city'];
$zip = $_POST['zip'];
$country = $_POST['country'];

// se tiver sido preenchido
if($_SESSION['cart']&&$name&&$address&&$city&&$zip&&$country)
{
  // está apto para ser inserido no banco de dados
  if( insert_order($_POST)!=false )
  {
    //exibe carrinho, não permitindo mudanças e sem figuras
    display_cart($_SESSION['cart'], false, 0);

    display_shipping(calculate_shipping_cost( ));

    //obtém os detalhes sobre cartão de crédito
    display_card_form($name);

    display_button('show_cart.php', 'continue-shopping', 'Continue Shopping');
  }
  else
  {
    echo 'Could not store data, please try again.';
    display_button('checkout.php', 'back', 'Back');
  }
}
else
{
  echo 'You did not fill in all the fields, please try again.<hr />';
  display_button('checkout.php', 'back', 'Back');
}

do_html_footer( );
?>
```

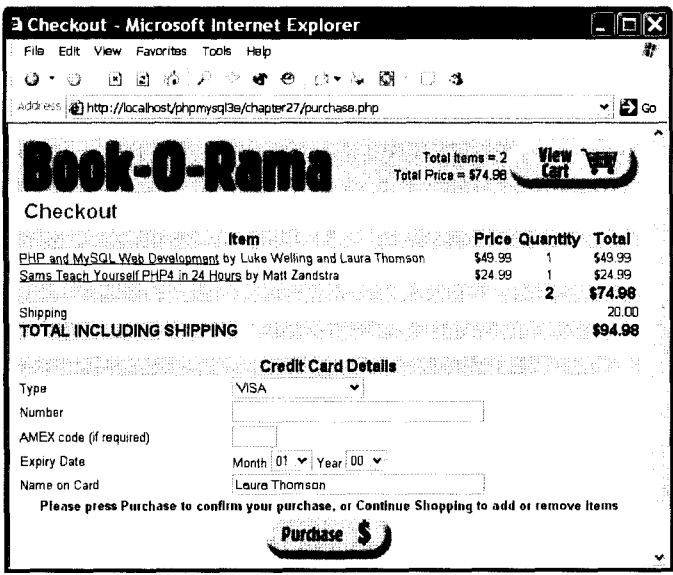



Figura 27.9 O script purchase.php calcula o custo de remessa e o total final do pedido e obtém os detalhes de pagamento do cliente.

Eis a lógica simples e direta: verificamos se o usuário preencheu o formulário e inseriu detalhes no banco de dados utilizando uma função chamada `insert_order( )`. Essa é uma função simples que coloca os detalhes do cliente no banco de dados. O código para essa função é mostrado na Listagem 27.15.

Listagem 27.15 **Função `insert_order( )` de `order_fns.php` – Essa função insere todos os detalhes do pedido do cliente no banco de dados**

```
function insert_order($order_details)
{
 // extrai detalhes do pedido como variáveis
 extract($order_details);

 // define endereço de entrega como o mesmo endereço
 if(!$ship_name&&!$ship_address&&!$ship_city&&!$ship_state&&
    !$ship_zip&&!$ship_country)
 {
   $ship_name = $name;
   $ship_address = $address;
   $ship_city = $city;
   $ship_state = $state;
   $ship_zip = $zip;
   $ship_country = $country;
 }

 $conn = db_connect( );

 // queremos inserir o pedido como uma transação
 // inicia uma desativando autocommit
 $conn->autocommit(FALSE);

 // insere endereço do cliente
 $query = "select customerid from customers where
           name = '$name' and address = '$address'
           and city = '$city' and state = '$state'
           and zip = '$zip' and country = '$country'";
 $result = $conn->query($query);
 if($result->num_rows>0)
 {
   $customer = $result->fetch_object( );
   $customerid = $customer->customerid;
 }
 else
 {
   $query = "insert into customers values
          ('', '$name','$address','$city','$state','$zip','$country')";
   $result = $conn->query($query);
   if (!$result)
      return false;
 }
 $customerid = $conn->insert_id;

 $date = date('Y-m-d');
 $query = "insert into orders values
          ('', $customerid, ".$_SESSION['total_price'].", '$date',
           'PARTIAL', '$ship_name',
           '$ship_address','$ship_city','$ship_state','$ship_zip',
            '$ship_country')";

 $result = $conn->query($query);
 if (!$result)
```

Listagem 27.15 **Continuação**

```
  return false;

$query = "select orderid from orders where
            customerid = $customerid and
            amount > ".$_SESSION['total_price']."-.001 and
            amount < ".$_SESSION['total_price']."+.001 and
            date = '$date' and
            order_status = 'PARTIAL' and
            ship_name = '$ship_name' and
            ship_address = '$ship_address' and
            ship_city = '$ship_city' and
            ship_state = '$ship_state' and
            ship_zip = '$ship_zip' and
            ship_country = '$ship_country'";
$result = $conn->query($query);
if($result->num_rows>0)
{
  $order = $result->fetch_object( );
  $orderid = $order->orderid;
}
else
  return false;

// insere cada livro
foreach($_SESSION['cart'] as $isbn => $quantity)
{
  $detail = get_book_details($isbn);
  $query = "delete from order_items where
           orderid = '$orderid' and isbn =  '$isbn'";
  $result = $conn->query($query);
  $query = "insert into order_items values
           ('$orderid', '$isbn', ".$detail['price'].", $quantity)";
  $result = $conn->query($query);
  if(!$result)
    return false;
}

// finaliza a transação
$conn->commit( );
$conn->autocommit(TRUE);

return $orderid;
}
```

A função `insert_order( )` é bastante longa porque você precisa inserir os detalhes do cliente, do pedido e de cada livro que ele deseja comprar.

Você pode notar que as diferentes partes da inserção ficam dentro de uma transação, começando com:

```
$conn->autocommit(FALSE);
```

e terminando com:

```
$conn->commit( );
$conn->autocommit(TRUE);
```

Esse é o único local da aplicação em que é necessário utilizar uma transação. Como evitar ter de usá-la em outro lugar? Observe o código na função `db_connect( )`:

```
function db_connect( )
{
  $result = new mysqli('localhost', 'book_sc', 'password', 'book_sc');
  if (!$result)
     return false;
  $result->autocommit(TRUE);
  return $result;
}
```

Obviamente, esse código é um pouco diferente daquele utilizado para essa função em outros capítulos. Depois de criar a conexão ao MySQL, ative o modo autocommit. Isso assegura que cada instrução SQL seja automaticamente concluída, como já vimos antes. Quando quiser na verdade utilizar uma transação multiinstrução, desative o autocommit, realize uma série de inserções, envie os dados e então ative novamente o modo autocommit.

Em seguida, calcule os custos de entrega para o endereço do cliente e informe quanto será com a seguinte linha de código:

```
display_shipping(calculate_shipping_cost( ));
```

O código utilizado aqui para `calculate_shipping_cost( )` sempre retorna $20. Ao configurar de verdade um site de compras, você deve escolher um método de entrega, descobrir o quanto custa a entrega para diferentes locais e calcular esses custos de acordo.

Depois, exiba um formulário para que o usuário preencha com os detalhes sobre o cartão de crédito utilizando a função `display_card_form( )` da biblioteca `output_fns.php`.

Implementando o pagamento

Quando o usuário clicar no botão Purchase, processaremos seus detalhes de pagamento usando o script `process.php`. Você pode ver os resultados de um pagamento bem-sucedido na Figura 27.10.

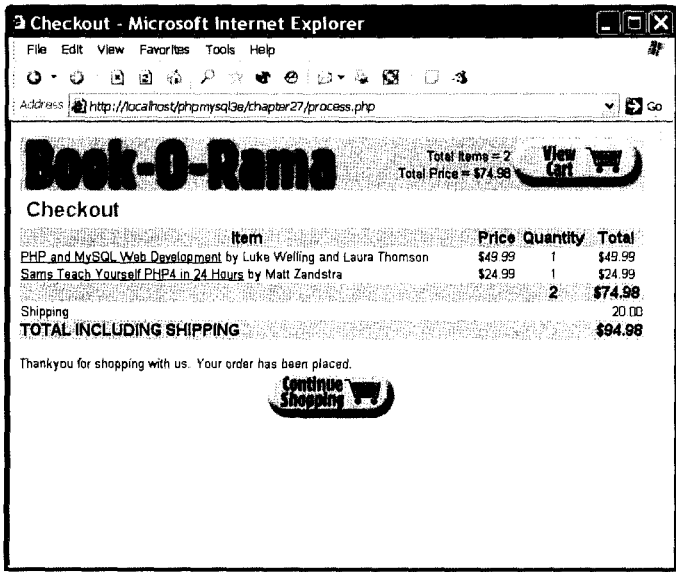



Figura 27.10 Essa transação foi bem-sucedida e os itens agora serão enviados.

O código para `process.php` pode ser encontrado na Listagem 27.16.

Listagem 27.16 **process.php – Script que processa o pagamento do cliente e informa-lhe o resultado**

```
<?php
 require ('book_sc_fns.php');
 // O carrinho de compras precisa de sessões, então inicie uma
 session_start( );

 do_html_header('Checkout');

 $card_type = $_POST['card_type'];
 $card_number = $_POST['card_number'];
 $card_month = $_POST['card_month'];
 $card_year = $_POST['card_year'];
 $card_name = $_POST['card_name'];

 if($_SESSION['cart']&&$card_type&&$card_number&&
    $card_month&&$card_year&&$card_name )
 {
   //exibe o carrinho, sem permitir mudanças e sem figuras
   display_cart($_SESSION['cart'], false, 0);

   display_shipping(calculate_shipping_cost( ));

   if(process_card($_POST))
   {
     //carrinho de compras vazio
     session_destroy( );
     echo 'Thank you for shopping with us.  Your order has been placed.';
     display_button('index.php', 'continue-shopping', 'Continue Shopping');
   }
   else
   {
   echo 'Could not process your card. ';
   echo 'Please contact the card issuer or try again.';
     display_button('purchase.php', 'back', 'Back');
   }
 }
 else
 {
   echo 'You did not fill in all the fields, please try again.<hr />';
   display_button('purchase.php', 'back', 'Back');
 }

 do_html_footer( );
?>
```

Processamos o cartão do usuário, e, se tudo for bem-sucedido, destruímos sua sessão.

A função de processamento de cartão da maneira como a escrevemos simplesmente retorna true. Se você realmente o estivesse implementando, precisaria realizar alguma validação (verificando se a data de validade e o número do cartão são válidos) e então processaria o pagamento real.

Quando configurar um site ao vivo, você precisará tomar uma decisão sobre qual mecanismo de compensação de transação quer utilizar. Você pode:

- Inscrever-se em um provedor de compensação de transação. Alguns desses oferecerão compensação em tempo real e outros não. Se você precisa ou não de compensação instantânea depende do serviço que oferece. Se estiver fornecendo um serviço on-line, você provavelmente deve querer isso; se estiver despachando a mercadoria, isso é menos crucial. De qualquer maneira, esses provedores aliviam você da responsabilidade de armazenar números de cartão de crédito.

- Enviar um número de cartão de crédito para você mesmo via e-mail encriptado, por exemplo, utilizando PGP ou GPG como abordado no Capítulo 17. Quando receber e decriptar o e-mail, você pode processar essas transações manualmente.
- Armazenar os números de cartão de crédito no seu banco de dados. Não recomendamos essa opção a menos que você saiba realmente o que está fazendo com segurança de sistema. Você pode ler o Capítulo 17 para obter mais detalhes da razão pela qual essa é uma má idéia.

Isso é tudo para os módulos de carrinho de compras e de pagamento.

Implementando uma interface de administração

A interface de administração que implementamos é muito simples. Tudo que fizemos foi construir uma interface Web para o banco de dados com um front-end de autenticação. Isso é muito parecido com o código que utilizamos no Capítulo 26. Isso foi incluído aqui apenas para sermos completos, mas com pouca discussão.

A interface de administração exige que um usuário efetue o login via arquivo login.php, que o leva até o menu de administração, admin.php. A página de login é mostrada na Figura 27.11. (Omitimos o arquivo login.php aqui para sermos breves – esse arquivo é quase idêntico ao arquivo do Capítulo 26. Se quiser vê-lo, está no CD-ROM.) O menu de administração é mostrado na Figura 27.12.

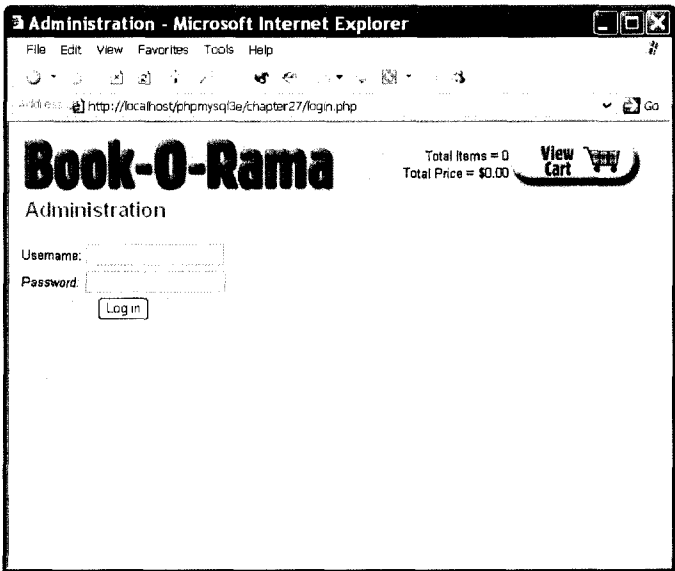



Figura 27.11 Os usuários devem passar pela página de login para acessar as funções de administração.

O código para o menu de administração é mostrado na Listagem 27.17.

Listagem 27.17 **admin.php – Esse script autentica o administrador e permite que ele acesse as funções de administração**

```
<?php

// inclui arquivos de função para essa aplicação
require_once('book_sc_fns.php');
session_start( );


if ($_POST['username'] && $_POST['passwd'])
```

Listagem 27.17 **Continuação**

```
// acabaram de tentar fazer logon
{

  $username = $_POST['username'];
  $passwd = $_POST['passwd'];

  if (login($username, $passwd))
  {
    // se estiver no banco de dados, registre o id do usuário
    $_SESSION['admin_user'] = $username;
  }
  else
  {
    // login malsucedido
    do_html_header('Problem:');
    echo 'You could not be logged in.
          You must be logged in to view this page.<br />';
    do_html_url('login.php', 'Login');
    do_html_footer( );
    exit;
  }
}

do_html_header('Administration');
if (check_admin_user( ))
 display_admin_menu( );
else
 echo 'You are not authorized to enter the administration area.';

do_html_footer( );

?>
```

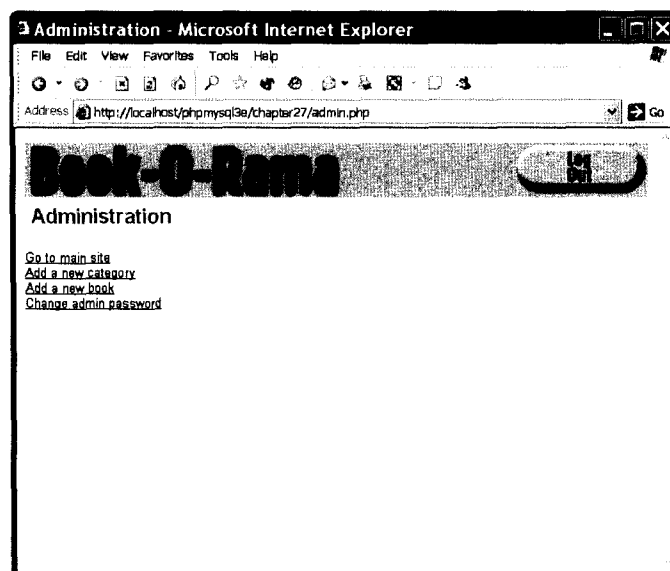



Figura 27.12 O menu de administração permite acessar as funções de administração.

Esse código provavelmente parece familiar; ele é semelhante a um script do Capítulo 26. Depois que o administrador alcançar esse ponto, ele pode alterar sua senha ou efetuar logout – esse código é idêntico ao código do Capítulo 26, então não o abordaremos aqui.

Identificamos o usuário de administração depois do login por meio da variável de sessão `admin_user` e da função `check_admin_user( )`. Essa função e as outras utilizadas pelos scripts administrativos podem ser localizadas na biblioteca de funções `admin_fns.php`.

Se o administrador escolher adicionar uma nova categoria ou livro, ele irá para `insert_category_form.php` ou para `insert_book_form.php`, conforme apropriado. Cada um desses scripts apresenta ao administrador um formulário para preenchimento. Cada um é processado por um script correspondente (`insert_category.php` e `insert_book.php`), que verifica se o formulário está preenchido e insere os novos dados no banco de dados. Veremos somente as versões de livro dos scripts, uma vez que elas são muito semelhantes entre si.

A saída de `insert_book_form.php` é mostrada na Figura 27.13.

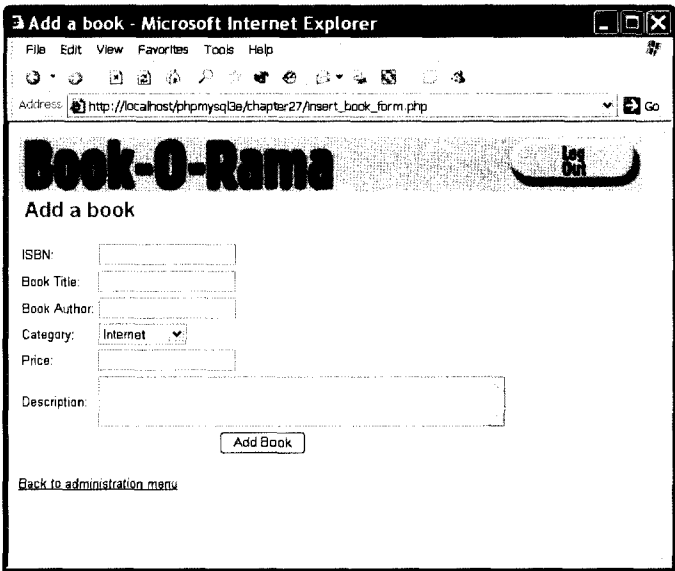



Figura 27.13 Esse formulário permite ao administrador inserir novos livros no catálogo on-line.

Observe que o campo `Category` para livros é um elemento HTML `SELECT`. As opções para esse elemento `SELECT` vêm de uma chamada para a função `get_categories( )` que vimos anteriormente.

Quando o botão Add Book for clicado, o script `insert_book.php` será ativado. O código para esse script é mostrado na Listagem 27.18.

Listagem 27.18 **`insert_book.php` – Esse script valida os novos dados de livro e os coloca no banco de dados**

```
<?php

// inclui arquivos de função para essa aplicação
require_once('book_sc_fns.php');
session_start( );

do_html_header('Adding a book');
if (check_admin_user( ))
{
  if (filled_out($_POST))
  {
    $isbn = $POST['isbn'];
    $title = $_POST['title'];
    $author = $_POST['author'];
    $catid = $_POST['catid'];
```

Listagem 27.18 Continuação

```
    $price = $_POST['price'];
    $description = $_POST['description'];

    if(insert_book($isbn, $title, $author, $catid, $price, $description))
      echo "Book '".stripslashes($title)."' was added to the database.<br />";
    else
      echo "Book '".stripslashes($title).
           "' could not be added to the database.<br />";
  }
  else
    echo 'You have not filled out the form.  Please try again.';
  do_html_url('admin.php', 'Back to administration menu');
}
else
  echo 'You are not authorised to view this page.';

do_html_footer( );

?>
```

Você pode ver que esse script chama a função `insert_book( )`. Essa função e as outras utilizadas pelos scripts administrativos podem ser encontradas na biblioteca de funções `admin_fns.php`.

Além de adicionar novas categorias e livros, o usuário administrativo pode editar e excluir esses itens. Implementamos isso reutilizando o máximo de código possível. Quando o administrador clicar no link Go to Main site no menu administration, ela irá para o índice de categorias em `index.php` e poderá navegar pelo site da mesma maneira que um usuário comum, utilizando os mesmos scripts.

Mas, há uma diferença na navegação administrativa: os administradores verão diferentes opções baseadas no fato de que eles têm a variável de sessão registrada `admin_user`. Por exemplo, se você examinar a página `show_book.php` que vimos anteriormente no capítulo, notará algumas opções diferentes de menu. Veja a Figura 27.14.

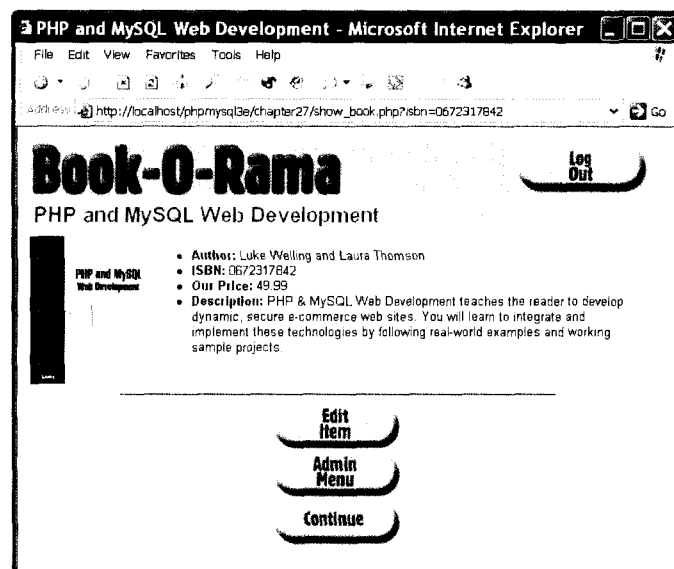



Figura 27.14 O script `show_book.php` produz saída diferente para um usuário administrativo.

O administrador tem acesso a duas novas opções nessa página: `Edit Item` e `Admin Menu`. Você também notará que não vemos o carrinho de compras no canto superior direito – em vez disso, temos um botão Log Out.

O código para isso está todo lá, de volta na Listagem 27.8, assim:

```
if( check_admin_user( ) )
{
  display_button("edit_book_form.php?isbn=$isbn", 'edit-item', 'Edit Item');
  display_button('admin.php', 'admin-menu', 'Admin Menu');
  display_button($target, 'continue', 'Continue');
}
```

Se examinar outra vez o script `show_cat.php`, você verá que ele também tem essas opções predefinidas para ele.

Se o administrador clicar no botão Edit Item, ele irá para o script `edit_book_form.php`. A saída desse script é mostrada na Figura 27.15.

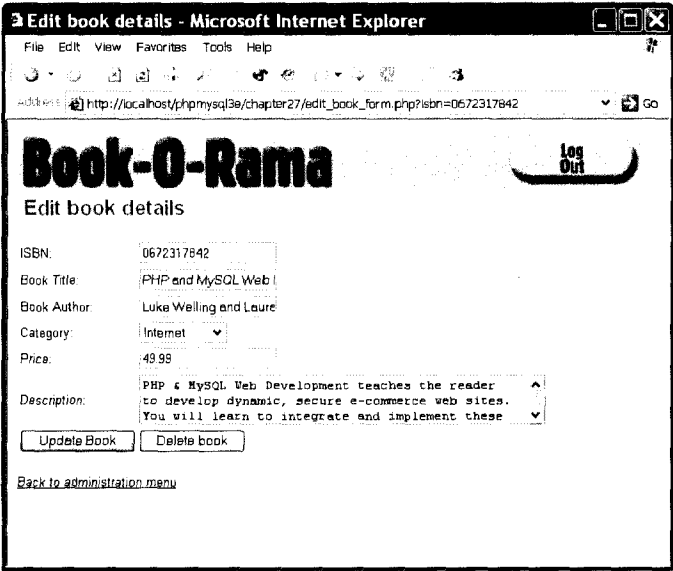



Figura 27.15 O script `edit_book_form.php` fornece acesso de administrador para editar detalhes de livro ou excluir um livro.

De fato, esse é o mesmo formulário que utilizamos para obter os detalhes do livro em primeiro lugar. Construímos uma opção nesse formulário para passar e exibir dados de livros existentes. Fizemos o mesmo com o formulário de categoria. Para ver o que estamos querendo dizer, examine a Listagem 27.19.

Listagem 27.19 **Função `display_book_form( )` de `admin_fns.php` – Esse formulário faz realmente um trabalho duplo como um formulário de inserção e edição**

```
function display_book_form($book = '')
// Isso exibe o formulário de livro.
// Ele é muito semelhante ao formulário de categorias.
// Esse formulário pode ser utilizado para inserir ou editar livros.
// Para inserir, não passe nenhum parâmetro. Isso configurará $edit
// como false e o formulário irá para insert_book.php.
// Para atualizar, passe um array contendo um livro. O formulário
// será exibido com os dados antigos e apontará para update_book.php.
// Ele também adicionará um botão "Delete book".
{
```

Listagem 27.19 **Continuação**

```
  // se passou um livro existente, prossegue no "modo de edição"
  $edit = is_array($book);

  // a maior parte do formulário está em HTML simples
  // fragmentos de código em PHP opcionais
?>
  <form method="post"
        action="<?php echo $edit?'edit_book.php':'insert_book.php';?>">
  <table border="0">
  <tr>
    <td>ISBN:</td>
    <td><input type="text" name="isbn"
         value="<?php echo $edit?$book['isbn']:''; ?>"></td>
  </tr>
  <tr>
    <td>Book Title:</td>
    <td><input type="text" name="title"
         value="<?php echo $edit?$book['title']:''; ?>"></td>
  </tr>
  <tr>
    <td>Book Author:</td>
    <td><input type="text" name="author"
         value="<?php echo $edit?$book['author']:''; ?>"></td>
   </tr>
  <tr>
      <td>Category:</td>
      <td><select name="catid">
      <?php
  // lista de possíveis categorias proveniente do banco de dados
          $cat_array=get_categories( );
          foreach ($cat_array as $thiscat)
          {
               echo '<option value="';
               echo $thiscat['catid'];
               echo '"';
               // se for livro existente, coloca-o na categoria atual
               if ($edit && $thiscat['catid'] == $book['catid'])
                   echo ' selected';
               echo '>';
               echo $thiscat['catname'];
               echo "\n";
          }
          ?>
          </select>
        </td>
   </tr>
   <tr>
    <td>Price:</td>
    <td><input type="text" name="price"
               value="<?php echo $edit?$book['price']:''; ?>"></td>
   </tr>
   <tr>
     <td>Description:</td>
     <td><textarea rows="3" cols="50"
          name="description">
          <?php echo $edit?$book['description']:''; ?>
          </textarea></td>
   </tr>
   <tr>
     <td <?php if (!$edit) echo 'colspan="2"'; ?> align="center">
```

Listagem 27.19 Continuação

```
        <?php
           if ($edit)
            // precisamos do isbn antigo para localizar o livro no banco de dados
            // se o isbn estiver sendo atualizado
            echo '<input type="hidden" name="oldisbn"
                   value="'.$book['isbn'].'">';
        ?>
        <input type="submit"
               value="<?php echo $edit?'Update':'Add'; ?> Book">
        </form></td>
        <?php
          if ($edit)
          {
            echo '<td>';
            echo '<form method="post" action="delete_book.php">';
            echo '<input type="hidden" name="isbn"
                   value="'.$book['isbn'].'">';
            echo '<input type="submit"
                   value="Delete book">';
            echo '</form></td>';
          }
         ?>
        </td>
      </tr>
 </table>
  </form>
<?php
}
```

Se passarmos um array contendo os dados de livro, o formulário será gerado no modo de edição e preencherá os campos com os dados existentes:

```
<input type="text" name="price"
               value="<?php echo $edit?$book['price']:''; ?>">
```

Obtemos ainda um botão de enviar diferente. De fato, para o formulário de edição obtemos dois – um para atualizar o livro e um para excluí-lo. Esses botões chamam os scripts `edit_book.php` e `delete_book.php`, que atualizam o banco de dados de maneira correspondente.

As versões de categoria desses scripts funcionam quase da mesma maneira, exceto por um item. Quando um administrador tentar excluir uma categoria, ela não será excluída se algum livro ainda estiver nela. (Isso é verificado com uma consulta de banco de dados.) Isso evita qualquer problema que poderíamos obter com anomalias de exclusão. Discutimos isso no Capítulo 8. Nesse caso, se uma categoria que ainda contém livros for excluída, esses livros se tornarão órfãos. Não saberíamos em que categoria eles estavam e não teríamos nenhuma maneira de navegar por eles!

Essa é a visão geral da interface de administração. Para detalhes adicionais, consulte o código – esse código está completo no CD-ROM.

Estendendo o projeto

Construímos um sistema relativamente simples de carrinho de compras. Há muitas adições e aprimoramentos que poderíamos fazer:

- Em uma loja on-line real, você precisaria construir algum tipo de sistema de monitoramento e atendimento de pedidos – no momento, não há nenhuma maneira de ver que pedidos foram feitos.

- Os clientes devem ser capazes de verificar o progresso de seus pedidos sem entrar em contato com você. Consideramos que é importante que o cliente não tenha de efetuar logon para navegar. Mas, fornecer aos clientes existentes uma maneira de se autenticar oferece a capacidade de ver os pedidos do passado e a capacidade de associar comportamentos a um perfil.
- Atualmente, as imagens dos livros precisam ser feitas em FTP para o diretório de imagem e precisam receber o nome correto. Você poderia adicionar upload de arquivo à página de inserção de livro para tornar isso mais fácil.
- Você poderia adicionar login de usuário, personalização e recomendações de livro; revisões on-line; programas de afiliação; verificação de nível do estoque; e assim por diante. As possibilidades são infinitas.

Utilizando um sistema existente

Se quiser obter um carrinho de compras altamente equipado, fácil de instalar e pronto para executar, talvez você deva tentar utilizar um sistema de carrinho de compras existente. Um famoso sistema de carrinho de compras de código-fonte aberto implementado em PHP é o FishCartSQL, disponível em:

http://www.fishcart.org/

Esse sistema tem uma grande quantidade de recursos avançados como monitoramento de cliente, vendas sincronizadas, vários idiomas, processamento de cartão de crédito e suporte para diversas lojas on-line em um servidor. Naturalmente, quando utiliza um sistema existente, você sempre descobre coisas que ele não tem mas que você quer e vice-versa. A vantagem de um produto de código-fonte aberto é que você pode entrar e alterar o que não gosta.

A seguir

No próximo capítulo, veremos como construir um sistema de gerenciamento de conteúdo on-line adequado para gerenciar recursos digitais – isso pode ser útil se você criar e manter um site baseado em conteúdo.

28

Construindo um sistema de gerenciamento de conteúdo

NESTE CAPÍTULO, VEREMOS UM SISTEMA DE GERENCIAMENTO de conteúdo para armazenar, indexar e pesquisar texto e conteúdo multimídia. Os sistemas de gerenciamento de conteúdo são extremamente úteis em Web sites cujo conteúdo é mantido por mais de um autor, em sites que a manutenção é realizada por pessoal não-técnico ou o conteúdo e o design gráfico são desenvolvidos por pessoas ou departamentos diferentes.

Neste capítulo, construiremos uma aplicação que ajuda usuários autorizados a gerenciar recursos digitais de uma empresa.

Esse capítulo aborda o seguinte:

- Como apresentar páginas Web utilizando uma série de modelos;
- Como construir um sistema de pesquisa que indexa documentos de acordo com metadados.

O problema

Vamos imaginar que a equipe ocupada com o desenvolvimento Web para o SuperFastOnlineNews consiste em um excelente designer gráfico e alguns escritores premiados. O site contém páginas regularmente atualizadas de notícias gerais, esportes e clima. A página principal mostra a última manchete de cada uma das três páginas de categoria.

No SuperFastOnlineNews, os designers asseguram que o conteúdo do Web site tenha uma excelente aparência. Isso é o que eles fazem de melhor. Os escritores, por outro lado, escrevem matérias excelentes, mas não podem desenhar bem nem construir Web sites.

Precisamos permitir que cada um se concentre no que é melhor e juntar sua saída para fornecer o superserviço de notícia rápida que o nome implica.

Requisitos da solução

Para esse projeto, precisamos produzir um sistema que:

- Aumente a produtividade tendo os escritores concentrados no texto e os designers no projeto;
- Permita ao editor revisar matérias e decidir quais devem ser publicadas;
- Apresente uma aparência e um comportamento consistentes por todo o site utilizando modelos de páginas;
- Permita aos escritores acessar somente suas áreas designadas do site;

- Permita que a aparência e o comportamento sejam facilmente alterados em uma seção ou por todo o site;
- Impeça que conteúdo ao vivo seja alterado.

Sistemas existentes

Existem muitos CMSs – tanto gratuitos como comerciais. Antes de escrever seu próprio, é melhor avaliar alguns existentes. A vantagem de usar o de alguém em vez de escrever o seu é similar a de outros projetos.

Escrever seu próprio CMS lhe garante total flexibilidade mas requer muito mais trabalho. Você pode decidir exatamente como a saída do CMS integrará no seu Web site e como o conteúdo dinâmico será tratado.

Sistemas existentes podem lhe fornecer recursos muito avançados com pouco trabalho. Eles geralmente têm saída muito flexível porque essa é uma das principais razões de ser do CMS, mas eles costumam deixá-lo preso a um determinado fluxo de trabalho e podem não lidar bem com conteúdo dinâmico.

Escrever um CMS simples – com os tipos de recursos que contruiremos neste capítulo – não leva muito tempo, mas quanto mais recursos você quiser, mais difícil será a tarefa e melhor será adotar um já existente.

Editando conteúdo

Primeiro, precisamos pensar sobre como obteremos o conteúdo no sistema e como armazenaremos e editaremos esse conteúdo.

Obtendo conteúdo para o sistema

Precisamos decidir sobre a maneira como as matérias e os componentes de design serão enviados. Três possíveis métodos podem ser utilizados.

FTP/SCP

Os escritores e designers poderiam ter acesso FTP ou SCP a áreas no servidor Web e poderiam, portanto, fazer uploads de arquivos de sua máquina local para o servidor. Você precisaria ter um padrão rígido de nomeação para arquivos transferidos por upload (de modo a identificar que imagens pertencem a quais matérias) ou um sistema baseado na Web para lidar com isso separadamente do upload de FTP.

Infelizmente, seria difícil fornecer direitos e permissões detalhados a diferentes usuários usando esses métodos; portanto, não o utilizamos neste exemplo.

Método de upload de arquivo

Como discutimos no Capítulo 18, o protocolo de HTTP fornece um método para que os arquivos sejam carregados via navegador Web. O PHP é capaz de lidar com isso muito facilmente.

O método de upload de arquivo também fornece a oportunidade de armazenar texto em um banco de dados em vez de como um arquivo. Para fazer isso, leríamos o arquivo temporário e armazenaríamos seu conteúdo no banco de dados, em vez de copiá-lo para outro lugar no sistema de arquivos. Não utilizaremos upload de arquivo para matérias nesse projeto.

Discutiremos a superioridade de um banco de dados sobre o sistema de arquivos mais adiante.

Editando on-line

Você pode deixar usuários criarem e editarem documentos sem utilizar FTP, SCP ou upload de arquivos. Em vez disso, pode oferecer aos colaboradores uma grande caixa de entrada de texto na tela, na qual o conteúdo da matéria poderia ser editado.

Esse método é simples mas geralmente eficaz. O navegador Web não fornece qualquer facilidade de edição de texto além da funcionalidade de cortar-e-colar do sistema operacional. No entanto, quando é preciso apenas fazer uma pequena mudança – por exemplo, corrigir um erro de grafia – pode pode rapidamente pegar o conteúdo e corrigi-lo. Infelizmente, como um `textarea` HTML não pode fornecer recursos avançados como correção ortográfica em tempo real, você provavelmente precisará corrigir os erros de grafia.

Semelhante ao upload de arquivos, os dados do formulário poderiam ser gravados em um arquivo ou armazenados em um banco de dados.

Armazenamento em bancos de dados *versus* armazenamento em arquivo

Uma decisão importante a tomar em uma etapa anterior é a maneira como o conteúdo será armazenado depois que foi carregado no sistema.

Como você armazenará metadados ao lado do texto da matéria, é possível inserir as partes do conteúdo de texto no banco de dados para esse projeto. Embora o MySQL seja capaz de armazenar dados multimídia, geralmente é melhor armazenar imagens transferidas por upload no sistema de arquivos, como fizemos aqui. Utilizar dados BLOB no banco de dados MySQL pode reduzir o desempenho, como vimos na Parte II.

Você pode apenas armazenar o nome de arquivo da imagem no banco de dados. Utilizando o sistema de arquivos, a tag `<IMG>` pode referenciar o arquivo de imagem diretamente como de costume.

Estrutura de documento

As matérias de exemplo que utilizaremos são notícias breves de um ou dois parágrafos com uma única imagem destinadas a pessoas com pressa. Elas são documentos estruturados uma vez que contêm uma manchete e um ou dois parágrafos de texto com uma imagem.

Quanto mais estruturado um documento é, mais facilmente ele poderá ser dividido para armazenamento em um banco de dados. A vantagem disso é que todos os documentos podem ser apresentados de maneira muito estruturada e consistente. A desvantagem correspondente é que mais estrutura leva a menos flexibilidade.

Tome nosso exemplo de uma matéria jornalística. Armazenaremos a manchete em um campo separado do texto da matéria; e, por sua natureza, a imagem é um componente separado do documento.

Com a manchete como um item separado, podemos definir que um estilo e uma fonte padrão sejam exibidos, e podemos facilmente separá-lo do restante da matéria para compor nossa página principal de manchetes.

Outra abordagem para documentos grandes seria ter um relacionamento um-para-muitos com os parágrafos individuais; isto é, armazenar cada parágrafo como uma linha separada no banco de dados, cada um vinculado a um ID de documento mestre. Esse tipo de estrutura dinâmica de documento permitiria apresentar uma página de conteúdo para cada documento e exibir cada seção independentemente ou exibir o documento inteiro de uma vez.

Utilizando metadados

Já decidimos que cada registro de matérias abrange uma manchete, o texto da matéria e uma imagem. Mas não há nenhuma razão de não podermos armazenar outros dados no mesmo registro.

Nosso sistema automaticamente inserirá valores para quem criou a matéria e para quando ela foi modificada pela última vez. Esses podem ser automaticamente exibidos na parte inferior de uma matéria para assiná-la e registrar sua data/hora sem que o autor precise se preocupar em adicionar essas informações.

Esse sistema poderia também ser útil para adicionar dados que não são exibidos, conhecidos como *metadados*. Um bom exemplo disso é armazenar palavras-chave que seriam utilizadas para o índice de sistema de pesquisa.

Em vez de varrer o texto inteiro de cada matéria, o sistema de pesquisa examinará os metadados de palavra-chave para cada matéria e determinará a relevância unicamente a partir daí. Isso permite ao administrador do site ter controle total sobre que palavras e frases de pesquisa correspondem a quais documentos.

Em nosso exemplo, permitiremos que qualquer número de palavras-chave seja associado a uma matéria e atribuiremos a cada palavra-chave um valor de peso para indicar o grau de relevância que ela tem em uma escala de 1 a 10.

Portanto, podemos desenvolver um algoritmo de sistema de pesquisa que classifica correspondências de acordo com essa relevância especificada por humano para matérias, em vez de um algoritmo complexo que tenha de interpretar o texto escrito em linguagem natural e tomar decisão com base em seu entendimento limitado e governado por regras fixas.

Se seus dados são basicamente texto armazenado em um banco de dados, a indexação e busca de texto completo do MySQL será uma solução melhor do que percorrer seu próprio sistema, mas o sistema desenvolvido aqui poderia ser utilizado para um mecanismo simples de busca utilizando diversos tipos de documento e arquivos multimídia.

Isso não é quer dizer que você seja obrigado a armazenar metadados no banco de dados. Não há nada que o impeça de utilizar a tag `<META>` em HTML ou mesmo de utilizar XML para construir seus documentos. Entretanto, sempre que você puder, vale tirar proveito do banco de dados que já está cuidando de seus documentos.

Formatando a saída

Nosso exemplo de site de notícias segue um formato simples mas estruturado de exibição de página. Cada página contém várias matérias, todas as quais são formatadas da mesma maneira. Primeiramente, a manchete é exibida em uma fonte grande, seguida pela fotografia abaixo e à esquerda e então pelo texto da matéria à direita. A página inteira é contida em um modelo padrão de página para preservar inteiramente a marca e a consistência do site.

A Figura 28.1 mostra a estrutura lógica de página que utilizaremos.

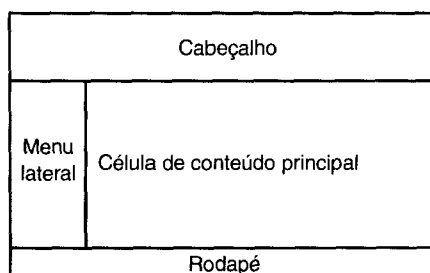



Figura 28.1 A estrutura lógica da página para as páginas públicas de SFON segue as convenções comuns e é fácil de produzir em HTML.

Não é difícil implementar uma estrutura simples com modelos como esse a partir de um projeto de página. Você simplesmente divide a página em três partes: cabeçalho, menu lateral e rodapé que não mudam e o conteúdo da página que varia de página a página e diariamente. Sempre que você

exibir uma página para esse site, mostra o cabeçalho e o menu lateral primeiro, depois o conteúdo da página e finalmente o rodapé.

O cabeçalho e o menu lateral estão em um arquivo (header.php) e o rodapé em outro (footer.php). O conteúdo principal de cada página é gerado por um script apropriado.

Implementar o site com um modelo de cabeçalho e rodapé permite que esses arquivos de modelo sejam facilmente modificados se o projeto do site for atualizado.

Projeto/visão geral da solução

Um resumo dos arquivos nessa aplicação é mostrado na Tabela 28.1.

Tabela 28.1 **Arquivos na aplicação de gerenciamento de conteúdo**

Nome	Tipo	Descrição
create_database.sql	SQL	SQL para configurar o banco de dados de conteúdo e alguns dados de exemplo.
include_fns.php	Funções	Coleção de arquivos include para essa aplicação.
db_fns.php	Funções	Coleção de funções para conectar-se ao banco de dados de conteúdo.
select_fns.php	Funções	Coleção de funções para auxiliar na criação de listas drop down <select> dos bancos de dados.
user_auth_fns.php	Funções	Coleção de funções para autenticar usuários admin.
header.php	Modelo	Manchete mostrada na parte superior de cada página de conteúdo.
footer.php	Modelo	Rodapé mostrado na parte inferior de cada página de conteúdo.
logo.gif	Imagem	O arquivo do logotipo exibido em header.php.
index.php	Aplicação	Resumo que mostra a manchete mais recente de cada página do site.
Admin/index.php	Aplicação	Menu para as funções administrativas para o site.
page.php	Aplicação	Lista as manchetes e o texto das matérias para uma página particular.
resize_image.php	Aplicação	Redimensiona uma imagem instantaneamente para headlines.php.
search_form.php	Aplicação	Formulário para inserir palavras-chave para pesquisar o conteúdo de site.
search.php	Aplicação	Exibe as manchetes de conteúdo para corresponder ao critério das palavras-chave.
login.php	Aplicação	Autentica a senha de um usuário e efetua logon do usuário no sistema.
logout.php	Aplicação	Efetua logout de um usuário no sistema.
writer.php	Aplicação	Lista de matérias que o usuário conectado escreveu com uma opção para adicionar, modificar ou excluir matérias.
story.php	Aplicação	A tela de detalhe da matéria para editar ou adicionar uma nova matéria.
story_submit.php	Aplicação	Adiciona nova matéria ou salva alterações de dados inseridos em story.php.
delete_story.php	Aplicação	Script que processa a solicitação de exclusão de matéria a partir de stories.php.

Tabela 28.1 **Continuação**

Nome	Tipo	Descrição
keywords.php	Aplicação	Lista as palavras-chave para uma matéria com opção de adicionar ou excluir palavras-chave.
keyword_add.php	Aplicação	Script que processa solicitação de adicionar palavra-chave a partir de keywords.php.
keyword_delete.php	Aplicação	Script que processa a solicitação de exclusão de palavra-chave a partir de keywords.php.
publish.php	Aplicação	Lista de matérias do editor que mostra as matérias que são publicadas com uma opção de alternar o status de cada.
publish_story.php	Aplicação	Script que processa uma solicitação de publicação a partir de publish.php.
unpublish_story.php	Aplicação	Processa uma solicitação de não-publicação a partir de publish.php.

Projetando o banco de dados

A Listagem 28.1 mostra a consulta SQL utilizada para criar o banco de dados para o sistema de conteúdo. Essa listagem é parte do arquivo create_database.sql. O arquivo no CD também contém consultas para preencher o banco de dados com alguns usuários e matérias de exemplo.

Listagem 28.1 **Trecho de create_database.sql – Arquivo de SQL para configurar o banco de dados de conteúdo**

```
drop database if exists content;

create database content;

use content;

drop table if exists writers;

create table writers (
  username   varchar(16) primary key,
  password   varchar(16) not null,
  full_name  text
);

drop table if exists stories;

create table stories (
  id         int primary key auto_increment,
  writer     varchar(16) not null,        # chave estrangeira writers.username
  page       varchar(16) not null,        # chave estrangeira pages.code
  headline   text,
  story_text text,
  picture    text,
  created    int,
  modified   int,
  published  int
);
drop table if exists pages;

create table pages (
  code       varchar(16) primary key,
```

Listagem 28.1 Continuação

```
  description text
);

drop table if exists writer_permissions;

create table writer_permissions (
  writer      varchar(16) not null,          # chave estrangeira writers.username
  page        varchar(16) not null           # chave estrangeira pages.code
);

drop table if exists keywords;

create table keywords (
  story       int not null,                  # chave estrangeira stories.id
  keyword     varchar(32) not null,
  weight      int not null
);

grant select, insert, update, delete
on content.*
to content@localhost identified by 'password';
```

Precisamos armazenar poucas informações sobre cada escritor, incluindo um nome de login e uma senha, na tabela `writers`. Armazenaremos seus nomes completos para exibir depois de cada matéria e para cumprimentá-los quando efetuarem logon.

A tabela pages contém o título para cada página nas quais as matérias podem ser exibidas. A tabela `writer_permissions` implementa um relacionamento de muitos-para-muitos para indicar para qual página um escritor pode enviar a matéria.

A tabela `stories` contém campos separados para `headline`, `story_text` e `picture` como discutido anteriormente. Os campos `created`, `modified` e `published` são campos de inteiro e armazenarão o valor de registro de data/hora do Unix nos horários relevantes.

Para criar o banco de dados, execute o seguinte comando:

```
mysql -u root < create_database.sql
```

Veja se você ainda não possui um banco de dados chamado `content`, porque senão este será substituído.

Implementando CMS

Agora que temos um banco de dados e uma função de redimensionar imagens a que recorrer, vamos nos ocupar da construção da principal parte do sistema.

O front-end

Vamos começar examinando `index.php`, mostrado na Listagem 28.2, que seria a primeira página que o visitante do site veria. Queremos mostrar suas manchetes da última matéria de cada página.

Listagem 28.2 **index.php – Script que mostra a manchete mais recente de cada página**

```
<?php
include_once('db_fns.php');
include_once('header.php');

$handle = db_connect( );
```

Listagem 28.2 Continuação

```
$pages_sql = 'select * from pages order by code';
$pages_result = $handle->query($pages_sql);

echo '<table border="0" width="400">';

while ($pages = $pages_result->fetch_assoc( ))
{
  $story_sql = "select * from stories
                where page = '{$pages['code']}'
                and published is not null
                order by published desc";

  $story_result = $handle->query($story_sql);

  if ($story_result->num_rows)
  {
    $story = $story_result->fetch_assoc( );
    echo "<tr>
          <td>
            <h2>{$pages['description']}</h2>
            <p>{$story['headline']}</p>
            <p align='right' class='morelink'>
              <a href='page.php?page={$pages['code']}'>
              Read more {$pages['code']} ...
              </a>
            </p>
          </td>
          <td width='100'>";
    if ($story['picture'])
    {
      echo '<img src="resize_image.php?image=';
      echo urlencode($story['picture']);
      echo '&max_width=80&max_height=60"  />';
    }
    echo '</td></tr>';
  }
}
echo '</table>';

include_once('footer.php');
?>
```

Esse script, como com todos os scripts públicos, inclui `header.php` no início e `footer.php` no final. Qualquer saída gerada pelo script, portanto, é exibida dentro da célula principal de conteúdo na página.

O trabalho pesado é feito por duas consultas de banco de dados. Primeiro,

```
select * from pages order by code
```

recuperará a lista de páginas que estão no banco de dados. Em seguida o conteúdo do loop

```
"select * from stories
where page = '".$pages['code']."'
and published is not null
order by published desc"
```

é executado para localizar as matérias nessa página em ordem inversa da data publicada.

A Figura 28.2 mostra a saída de `headline.php` utilizando os dados de exemplo da aplicação.

Figura 28.2 O script `headline.php` mostra as manchetes de cada página dentro do site.

Próximo de cada manchete, um link é gerado na seguinte forma:

```
<p align='right' class='morelink'>
  <a href='page.php?page=news'>
  Read more news ...
  </a>
</p>
```

Esse link é gerado dentro do loop precedente de modo que o valor da string da consulta da página e o nome da página sejam impressos junto com a manchete relevante. Clicar nesse link leva o visitante para `page.php` – a lista completa de matérias para a página particular. A fonte para `page.php` pode ser localizada na Listagem 28.3.

Listagem 28.3 **page.php – Script que mostra uma matéria ou todas as matérias publicadas para uma página**

```
<?php
 if (!isset($_REQUEST['page'])&&!isset($_REQUEST['story']))
 {
   header('Location: index.php');
   exit;
 }

 $page = $_REQUEST['page'];
 $story = intval($_REQUEST['story']);

 include_once('db_fns.php');
 include_once('header.php');

 $handle = db_connect( );
 if($story)
 {
   $query = "select * from stories
             where id = '$story' and
                   published is not null";
 }
 else
```

Listagem 28.3 Continuação

```
{
  $query = "select * from stories
            where page = '$page' and
                  published is not null
            order by published desc";
}
$result = $handle->query($query);

while ($story = $result->fetch_assoc( ))
{
  // manchete
  echo "<h2>{$story['headline']}</h2>";
  //figura
  if ($story['picture'])
  {
    echo '<div style="float:right; margin:0px 0px 6px 6px;">';
    echo '<img src="resize_image.php?image=';
    echo urlencode($story[picture]);
    echo '&max_width=200&max_height=120"  align = right/></div>';
  }
  // crédito
  $w = get_writer_record($story['writer']);
  echo '<br /><p class="byline">';
  echo $w[full_name].', ';
  echo date('M d, H:i', $story['modified']);
  echo '</p>';
  // texto principal
  echo $story['story_text'];
}
include_once('footer.php');
?>
```

Note que `page.php` requer um valor para a página ou um valor para a matéria. No caso de `page.php` ser chamada diretamente sem a string de consulta, a primeira condição

```
if (!isset($_REQUEST['page'])&&!isset($_REQUEST['story']))
{
 header('Location: index.php');
 exit;
}
```

enviará o visitante de volta para a página da manchete para que a omissão da página não cause um erro.

A primeira consulta é usada se queremos chamar essa página para exibir uma única matéria:

```
select * from stories
where id = '$story' and
     published is not null
```

A segunda consulta é usada para recuperar matérias para uma determinada página:

```
select * from stories
where page = '$page' and
     published is not null
order by published desc
```

A publicada mais recentemente é recuperada primeiro. Dentro de cada loop, a imagem carregada e a matéria são impressas na tela, junto com o nome do autor e a data da última modificação.

A Figura 28.3 mostra page.php em ação, exibindo todos os itens da página de notícias para a aplicação de exemplo.

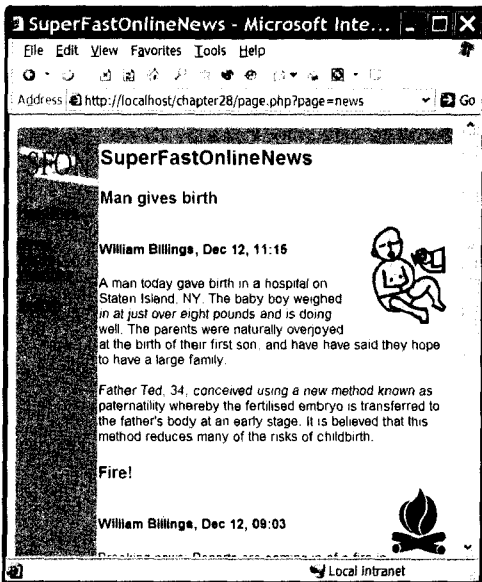



Figura 28.3 O script page.php mostra todas as matérias publicadas na página de notícias.

Manipulação de imagem

Os escritores colaboradores provavelmente fornecerão suas próprias fotografias para complementar a matéria. Queremos consistência, mas o que acontece quando um escritor carrega uma grande imagem de alta de qualidade e o outro carrega uma imagem miniatura?

Supondo que as figuras em questão são principalmente fotografias, podemos insistir somente em imagens de JPEG e tirar proveito de funções no PHP para manipular as imagens. Esse tópico é abordado em detalhe no Capítulo 21.

Criamos um script simples chamado resize_image.php, mostrado na Listagem 28.4, que você pode utilizar para redimensionar uma imagem instantaneamente de modo que possa ser exibida com uma tag <img> em um tamanho consistente. Redimensionar imagens instantaneamente pode não ser apropriado para um site muito procurado porque será uma operação cara. Usar isso nessa aplicação permite ter flexibilidade de exibir a mesma imagem em diferentes tamanhos em diferentes páginas – nesse caso, imagens menores na página da manchete e imagens maiores em páginas específicas. Também significa que é possível mudar o tamanho das imagens para ajustar um novo layout se você mudar o modelo.

Listagem 28.4 **resize_image.php – Script que redimensiona uma imagem JPEG instantaneamente**

```
<?php

 $image = $_REQUEST['image'];
 $max_width = $_REQUEST['max_width'];
 $max_height = $_REQUEST['max_height'];

 if (!$max_width)
   $max_width = 80;
 if (!$max_height)
   $max_height = 60;
```

Listagem 28.4 **Continuação**

```
  $size = GetImageSize($image);
  $width = $size[0];
  $height = $size[1];

  $x_ratio = $max_width / $width;
  $y_ratio = $max_height / $height;

  if ( ($width <= $max_width) && ($height <= $max_height) ) {
    $tn_width = $width;
    $tn_height = $height;
  }
  else if (($x_ratio * $height) < $max_height) {
    $tn_height = ceil($x_ratio * $height);
    $tn_width = $max_width;
  }
  else {
    $tn_width = ceil($y_ratio * $width);
    $tn_height = $max_height;
  }

  $src = ImageCreateFromJpeg($image);
  $dst = ImageCreate($tn_width,$tn_height);
  ImageCopyResized($dst, $src, 0, 0, 0, 0,
      $tn_width,$tn_height,$width,$height);
  header('Content-type: image/jpeg');
  ImageJpeg($dst, null, -1);
  ImageDestroy($src);
  ImageDestroy($dst);

?>
```

Esse script aceita três parâmetros – o nome de arquivo da imagem a exibir, a largura e a altura máximas em pixels. Isso não quer dizer que se o tamanho máximo especificado for 200×200, a imagem terá uma proporção de 200×200. Ao contrário, ela será reduzida proporcionalmente para que a maior largura ou altura seja de 200 pixels e a outra dimensão seja de 200 pixels ou menor. Por exemplo, uma imagem de 400×300 seria reduzida a 200×150. Dessa maneira, as proporções da imagem seriam mantidas.

Redimensionar instantaneamente no servidor é melhor opção do que simplesmente especificar os atributos `height` e `width` para a tag `<img>`. A grande imagem de alta resolução que um escritor enviou talvez seja de vários megabytes em tamanho; mas quando redimensionada para um tamanho razoável, poderia ficar abaixo de 100K. Portanto, não há nenhuma necessidade de fazer download do enorme arquivo e pedir ao navegador para redimensioná-lo.

As funções de manipulação de imagem são abordadas em detalhe no Capítulo 21. Aqui estamos utilizando a função `ImageCopyResized( )` a fim de escalar a imagem instantaneamente para o tamanho requerido.

A chave para a operação de redimensionamento é o cálculo dos novos parâmetros de largura e altura. Localizamos a relação entre as dimensões máximas e as reais. `$max_width` e `$max_height` podem ser passadas na string de consulta; caso contrário, os valores padrão especificados na parte superior da listagem serão utilizados.

```
$x_ratio = $max_width / $width;
$y_ratio = $max_height / $height;
```

Se a imagem já for menor que as dimensões máximas especificadas, a largura e a altura permanecem inalteradas. Caso contrário, a taxa de X ou Y é então utilizada para modificar as duas dimensões igualmente para que a imagem de tamanho reduzido não seja estendida nem comprimida, assim:

```
if ( ($width <= $max_width) && ($height <= $max_height) ) {
  $tn_width = $width;
  $tn_height = $height;
}
else if (($x_ratio * $height) < $max_height) {
  $tn_height = ceil($x_ratio * $height);
  $tn_width = $max_width;
}
else {
  $tn_width = ceil($y_ratio * $width);
  $tn_height = $max_height;
}
```

Depois de calculado o tamanho desejado, a imagem é redimensionada e enviada ao navegador. Para funcionar, esse script é chamado de dentro das tags `<img>` em uma página, e envia sua saída diretamente ao navegador com uma diretiva de cabeçalho apropriada.

Uma vantagem potencial dessa abordagem que não utilizamos aqui é que as imagens não precisam estar dentro da árvore de diretórios Web. Isso pode ter importantes vantagens em relação à segurança para uma aplicação na qual você permite que um script administrativo grave nos diretórios de imagem. Como o script passa a imagem, apenas ele precisa estar dentro da árvore Web.

O back-end

A seguir, vamos examinar como as matérias podem ser adicionadas ao sistema. Os escritores iniciam em `stories.php`. Esse script, depois que um escritor é autenticado, exibe uma lista de matérias que o autor escreveu e exibe a data de publicação ou oferece opções para adicionar uma nova matéria, editar ou excluir uma existente ou configurar as palavras-chave de pesquisa. Um exemplo é mostrado na Figura 28.4.

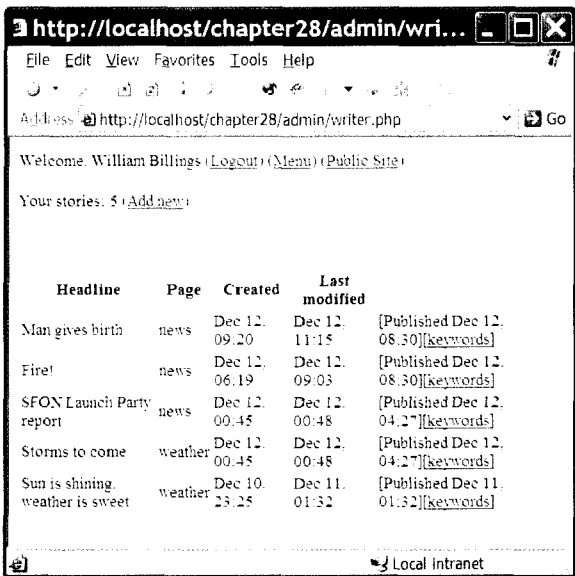



Figura 28.4 O script `writer.php` exibe a página de gerenciamento de matérias para escritores.

Essas telas não estão formatadas dentro de arquivos de cabeçalho e rodapé, embora pudessem estar se fosse desejado. Como somente os escritores e o editor utilizarão esses scripts, escolhemos em nosso exemplo formatar apenas o necessário para criar um sistema utilizável. O código para `stories.php` é mostrado na Listagem 28.5.

Listagem 28.5 **writer.php – A interface para que os autores gerenciem suas matérias**

```
<?php
 // writer.php é a interface para que os autores gerenciem suas matérias

 include_once('include_fns.php');

 if (!check_auth_user( ))
 {
   login_form( );
 }
 else
 {
   $handle = db_connect( );

   $writer = get_writer_record($_SESSION['auth_user']);

   echo '<p>Welcome, '.$writer['full_name'];
   echo ' (<a href="logout.php">Logout</a>)
         (<a href="index.php">Menu</a>)
         (<a href="../">Public Site</a>) </p>';
   echo '<p>';

   $query = 'select * from stories where writer = \''.
          $_SESSION['auth_user'].'\' order by created desc';
   $result = $handle->query($query);

   echo 'Your stories: ';
   echo $result->num_rows;
   echo ' (<a href="story.php">Add new</a>)';
   echo '</p><br /><br />';

   if ($result->num_rows)
   {
     echo '<table>';
     echo '<tr><th>Headline</th><th>Page</th>';
     echo '<th>Created</th><th>Last modified</th></tr>';
     while ($stories = $result->fetch_assoc( ))
     {
       echo '<tr><td>';
       echo $stories['headline'];
       echo '</td><td>';
       echo $stories['page'];
       echo '</td><td>';
       echo date('M d, H:i', $stories['created']);
       echo '</td><td>';
       echo date('M d, H:i', $stories['modified']);
       echo '</td><td>';
       if ($stories['published'])
       {
         echo '[Published '.date('M d, H:i', $stories['published']).']';
       }
       else
       {
         echo '[<a href="story.php?story='.$stories['id'].'">edit</a>] ';
         echo '[<a href="delete_story.php?story='.$stories['id'].
              '">delete</a>] ';
       }
       echo '[<a href="keywords.php?story='.$stories['id'].'">keywords</a>]';
       echo '</td></tr>';
     }
     echo '</table>';
   }
 }
?>
```

O primeiro passo serve para verificar se um usuário foi autenticado, e se não foi, serve para exibir somente um formulário de login.

A variável de sessão auth_user será configurada depois que um escritor tiver efetuado logon. A autenticação aqui não é particularmente segura e na realidade você tomaria mais cuidado para assegurar que os escritores fossem adequadamente autenticados. Isso é tratado em detalhe no Capítulo 16.

O formulário de login envia para login.php, que verifica o nome de usuário e a senha em relação aos valores de banco de dados. Se o login for bem-sucedido, o usuário é retornado à página de onde veio, utilizando o valor HTTP_REFERER. Isso significa que o script de login pode ser invocado de qualquer página de chamada dentro do sistema.

Em seguida, cumprimentamos o escritor pelo nome e oferecemos a oportunidade de efetuar logout. Esse link sempre aparecerá na parte superior de stories.php, então o usuário pode facilmente efetuar logout quando terminar.

```
$writer = get_writer_record($_SESSION['auth_user']);

echo '<p>Welcome, '.$writer['full_name'];
echo ' (<a href="logout.php">Logout</a>)
      (<a href="index.php">Menu</a>)
      (<a href="../">Public Site</a>) </p>';
```

A função get_writer_record( ) é definida em db_fns.php e retorna um array de todos os campos na tabela de escritor para o nome de usuário passado. O script logout.php simplesmente redefine o valor de auth_user.

A seguinte SQL localiza todas as matérias de um escritor, iniciando com a matéria adicionada mais recentemente:

```
$query = 'select * from stories where writer = \''.
       $_SESSION['auth_user'].'\' order by created desc';
```

Você armazena registros de data/hora criados, modificados e publicados em cada registro de matéria. Quando uma nova matéria é adicionada, os registros de data/hora criados e modificados são definidos para a hora do sistema. Cada mudança subseqüente atualiza apenas o campo modificado. Não defina o registro de data/hora publicado até que o editor publique a matéria para o site ao vivo.

Todas essas informações são mostradas na tela de matérias, primeiro com

```
echo date('M d, H:i', $stories['created']);
```

depois

```
echo date('M d, H:i', $stories['modified']);
```

e finalmente

```
if ($stories['published'])
{
  echo '[Published '.date('M d, H:i', $stories['published']).']';
}
else
{
  echo '[<a href="story.php?story='.$stories['id'].'">edit</a>] ';
  echo '[<a href="delete_story.php?story='.$stories['id'].'">delete</a>] ';
}
echo '[<a href="keywords.php?story='.$stories['id'].'">keywords</a>]';
```

Esse código mostra a data publicada, se apropriado; caso contrário, mostra os links para editar ou excluir aquela matéria. É possível adicionar palavras-chave de busca a matérias publicadas ou não.

O script para inserir uma nova matéria ou editar uma existente é story.php. A Figura 28.5 mostra uma das matérias sendo editada no banco de dados modelo da aplicação.

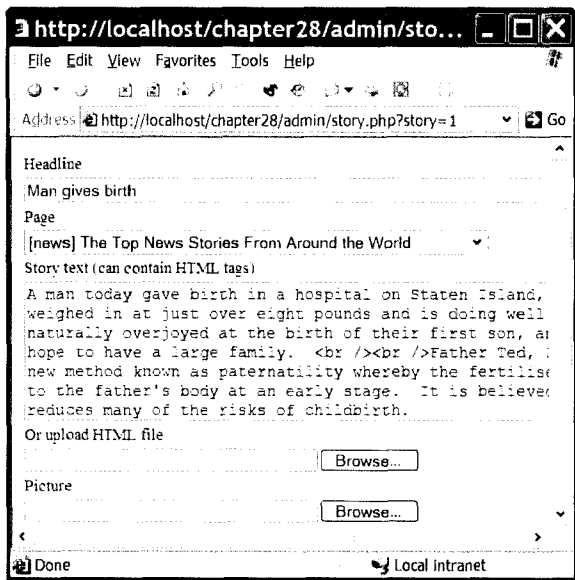



Figura 28.5 O script story.php permite que você edite a matéria.

A listagem completa de story.php pode ser vista na Listagem 28.6.

Listagem 28.6 **story.php – É utilizado para criar ou editar uma matéria**

```
<?php
  include ('include_fns.php');

  if (isset($_REQUEST['story']))
  {
    $story = get_story_record($_REQUEST['story']);
  }
?>

<form action="story_submit.php" method="post" enctype="multipart/form-data">
<input type="hidden" name="story" value="<?php echo $_REQUEST['story'];?>">
<input type="hidden" name="destination"
       value="<?php echo $_SERVER['HTTP_REFERER'];?>">
<table>

<tr>
  <td>Headline<td>
</tr>
<tr>
  <td><input size="80" name="headline"
            value="<?php echo $story['headline'];?>"></td>
</tr>

<tr>
  <td>Page</td>
</tr>
<tr>
  <td>
<?php
  if(isset($_REQUEST['story']))
```

Listagem 28.6 **Continuação**

```
  {
    $query = "select p.code, p.description
              from pages p, writer_permissions wp, stories s
              where p.code = wp.page
                    and wp.writer = s.writer
                    and s.id =".$_REQUEST['story'];
  }
  else
  {
    $query = "select p.code, p.description
              from pages p, writer_permissions wp
              where p.code = wp.page
                    and wp.writer = '{$_SESSION['auth_user']}'";
  }
  echo query_select('page', $query, $story['page']);
? >
  </td>
</tr>

<tr>
  <td>Story text (can contain HTML tags)</td>
</tr>
<tr>
  <td><textarea cols="80" rows="7" name="story_text"
          wrap="virtual"><?php echo $story['story_text'];?></textarea>
  </td>
</tr>

<tr>
  <td>Or upload HTML file</td>
</tr>
<tr>
  <td><input type="file" name="html" size="40"></td>
</tr>

<tr>
  <td>Picture</td>
</tr>
<tr>
  <td><input type="file" name="picture" size="40"></td>
</tr>

<?php
  if ($story[picture])
  {
    $size   = getImageSize('../'.$story['picture']);
    $width  = $size[0];
    $height = $size[1];
?>
    <tr>
      <td>
        <img src="<?php echo '../'.$story['picture'];?>"
              width="<?php echo $width;?>" height="<?php echo $height;?>">
      </td>
    </tr>
<?php
  }
?>

<tr>
```

Listagem 28.6 **Continuação**

```
  <td align="center"><input type="submit" value="Submit"></td>
</tr>
</table>
</form>
```

O mesmo script pode ser utilizado tanto para adição como para edição, e a ação depende de a matéria estar ou não configurada quando o script for chamado.

```
if (isset($_REQUEST['story']))
{
  $story = get_story_record($_REQUEST['story']);
}
```

A função `get_story_record( )` é definida em `db_fns.php` e retorna um array de todos os campos na tabela de matérias para o ID especificado de matéria. Se nenhum ID de matéria for passado, `$story` será nulo e `$s` não conterá os elementos do array.

```
<input size="80" name="headline"
       value="<?php echo $story['headline'];?>"></td>
```

Se `story` não estiver configurado, o código precedente não produzirá nenhum valor a partir da instrução de PHP, então a caixa de entrada de manchete estará em branco. Se `story` estiver configurado, ele conterá o texto de manchete para a matéria que está sendo editada.

A função `query_select( )` é definida em `select_fns.php` e retorna o código de HTML para produzir uma lista de `select` a partir de uma determinada consulta de SQL. O primeiro parâmetro é o atributo `name` para `select`. A consulta de SQL no segundo parâmetro seleciona duas colunas, onde a primeira é a parte `value` de cada opção e a segunda aparece depois da tag option e é o texto realmente exibido na lista. O terceiro parâmetro é opcional. Esse parâmetro adiciona um atributo `selected` à opção cujo valor corresponde ao valor especificado. Usamos a função para gerar um `select` que contenha as páginas para as quais esse escritor tem permissão de contribuir.

Para editar um artigo com o mesmo ID que ele tinha antes, é necessário gravar esse ID. Fazemos isso como uma variável escondida:

```
<input type="hidden" name="story" value="<?php print $_REQUEST['story'];?>">
```

Isso configura uma variável marcadora de lugar, configurando o novo valor para a matéria a partir da `story` passada. Quando o formulário é enviado, `story_submit.php` verifica se há um valor para `story` e gera uma instrução de SQL `UPDATE` ou `INSERT` correspondente.

O código para `story_submit.php` é mostrado na Listagem 28.7.

Listagem 28.7 **`story_submit.php` – Script utilizado para inserir ou atualizar uma matéria no banco de dados**

```
<?php
  // matéria_submit.php
  // adiciona / modifica o registro da matéria

  include_once('include_fns.php');

  $handle = db_connect( );

  $headline = $_REQUEST['headline'];
  $page = $_REQUEST['page'];
  $time = time( );

  if ( (isset($_FILES['html']['name']) &&
```

Listagem 28.7 **Continuação**

```
      (dirname($_FILES['html']['type']) == 'text') &&
      is_uploaded_file($_FILES['html']['tmp_name'])))
 {
   $story_text = file_get_contents($_FILES['html']['tmp_name']);
 }
 else
 {
   $story_text = $_REQUEST['story_text'];
 }

 $story_text = addslashes($story_text);

 if (isset($_REQUEST['story']) && $_REQUEST['story']!='')
 {   // É uma atualização
   $story = $_REQUEST['story'];

   $query = "update stories
             set headline = '$headline',
                 story_text = '$story_text',
                 page = '$page',
                 modified = $time
             where id = $story";
 }
 else
 {         // É uma nova matéria
   $query = "insert into stories
               (headline, story_text, page, writer, created, modified)
             values
               ('$headline', '$story_text', '$page', '".
              $_SESSION['auth_user']."', $time, $time)";
 }

 $result = $handle->query($query);

 if (!$result)
 {
   echo "There was a database error when executing <pre>$query</pre>";
   echo mysqli_error( );
   exit;
 }

 if ( (isset($_FILES['picture']['name']) &&
       is_uploaded_file($_FILES['picture']['tmp_name'])))
 {

   if (!isset($_REQUEST['story']) || $_REQUEST['story']=='')
   {
     $story = mysqli_insert_id($handle);
   }
   $type = basename($_FILES['picture']['type']);

   switch ($type) {
     case 'jpeg':
     case 'pjpeg':   $filename = "images/$story.jpg";
                     move_uploaded_file($_FILES['picture']['tmp_name'],
                                        '../'.$filename);
                     $query = "update stories
                               set picture = '$filename'
                               where id = $story";
                     $result = $handle->query($query);
```

Listagem 28.7 **Continuação**

```
                break;
      default:  echo 'Invalid picture format: '.
                     $_FILES['picture']['type'];
    }
  }

  header('Location: '.$_REQUEST['destination']);
?>
```

O link de exclusão de matéria chama `delete_story.php`, que aciona uma instrução `delete` simples e leva o escritor de volta para a página de chamada. O código para `delete_story.php( )` é mostrado na Listagem 28.8.

Listagem 28.8 **`delete_story.php` – Script utilizado para excluir uma matéria do banco de dados**

```
<?php
  // exclui_story.php

  include_once('include_fns.php');

  $handle = db_connect( );

  $story = $_REQUEST['story'];
  if(check_permission($_SESSION['auth_user'], $story))
  {
    $query = "delete from stories where id = $story";
    $result = $handle->query($query);
  }
  header('Location: '.$_SERVER['HTTP_REFERER']);
?>
```

Observe que a função `check_permission( )` se refere ao banco de dados para ver se o usuário conectado possui permissão para excluir matérias dessa página. Você usa essa função em muitas páginas na seção admin. Embora usuários não-autorizados não serão expostos aos links para excluir artigos, não seria bom se eles pudessem burlar a segurança simplesmente advinhando um URL óbvio.

Pesquisando

Clicar em link de palavras-chave na lista de matérias abre um formulário novo para inserir palavras-chave sobre a matéria. Não há nenhum limite para o número de palavras-chave que pode ser inserido, e cada palavra-chave recebe um valor de peso, com um valor mais alto indicando que ele é mais relevante.

A Figura 28.6 mostra a tela utilizada para configurar palavras-chave contra uma matéria particular.

O script `keywords.php` é relativamente simples e direto, então o examinaremos em detalhe. Ele está incluído no CD-ROM. Esse script desencadeia os scripts `keyword_add.php` e `keyword_delete.php`. Esses scripts também são simples e diretos e, portanto, não estão incluídos aqui.

O script `keyword_add.php` utiliza a seguinte consulta para adicionar novas palavras-chave ao banco de dados:

```
insert into keywords (story, keyword, weight)
values ($story, '$keyword', $weight)
```

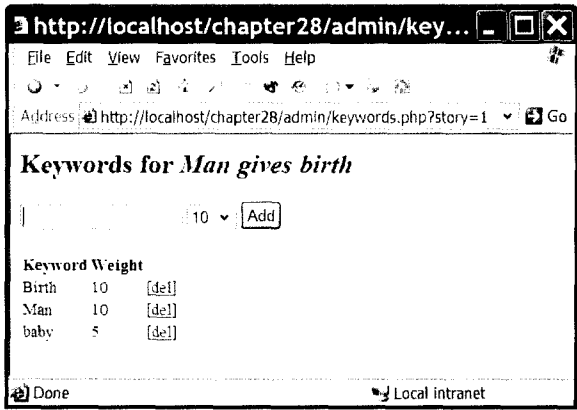



Figura 28.6 Configurando palavras-chave para uma matéria.

De maneira semelhante,`keyword_delete.php` utiliza a seguinte consulta para remover uma palavra-chave:

```
delete from keywords where story = $story and keyword = '$keyword'
```

O que é interessante é a maneira como os valores de peso são utilizados para calcular uma porcentagem de relevância ao pesquisar.

O formulário de pesquisa em `search_form.php` contém um único campo para palavras-chave e envia para `search.php`, que consulta o banco de dados de matérias ao vivo para localizar conteúdo correspondente. O código-fonte de `search.php` é mostrado na Listagem 28.9.

Listagem 28.9 **`search.php` – Script que encontra matérias correspondentes e calcula uma porcentagem de correspondência**

```
<?php
  include_once('db_fns.php');
  include_once('header.php');

  $handle = db_connect( );

  if ($_REQUEST['keyword'])
  {
    $keywords = explode(' ', $_REQUEST['keyword']);
    $num_keywords = count($keywords);
    for ($i=0; $i<$num_keywords; $i++)
    {
      if ($i)
      {
        $keywords_string .= "or k.keyword = '".$keywords[$i]."' ";
      }
      else
      {
        $keywords_string .= "k.keyword = '".$keywords[$i]."' ";
      }
    }

    $query = "select s.id,
                     s.headline,
                     10 * sum(k.weight) / $num_keywords as score
              from stories s, keywords k
              where s.id = k.story
                    and ($keywords_string)
                    and published is not null
```

Listagem 28.9 **Continuação**

```
            group by s.id, s.headline
            order by score desc, s.id desc";


    $result = $handle->query($query);
  }
  echo '<h2>Search results</h2>';

  if ($result  && $result->num_rows)
  {
    echo '<table>';
    while ($matches = $result->fetch_assoc( ))
    {
      echo "<tr><td><a href='page.php?story={$matches['id']}'>
             {$matches['headline']}
             </td><td>";
      echo floor($matches['score']).'%';
      echo '</td></tr>';
    }
    echo '</table>';
  }
  else
  {
    echo 'No matching stories found';
  }
  include_once('footer.php');
?>
```

Primeiro, a string de palavra-chave passada ao script `search.php` é dividida em palavras individuais de pesquisa. Não estamos utilizando nenhuma técnica de busca avançada neste exemplo, como permitir que o pesquisador utilize palavras-chave `AND` ou `OR` ou agrupe as palavras em uma expressão; portanto, todas as palavras na string serão palavras-chave.

```
if ($_REQUEST['keyword'])
{
  $keywords = split(' ', $_REQUEST['keyword']);
  $num_keywords = count($keywords);
  for ($i=0; $i<$num_keywords; $i++)
  {
    if ($i)
    {
      $keywords_string .= "or k.keyword = '".$keywords[$i]."' ";
    }
    else
    {
      $keywords_string .= "k.keyword = '".$keywords[$i]."' ";
    }
  }
```

Esse código utiliza a função do PHP `explode( )` para criar um array contendo cada palavra na string de palavra-chave separada por um caractere de espaço. Se apenas uma palavra for especificada, ela retorna uma único elemento de array e o loop subseqüente é executado uma vez. Finalmente, a condição armazenada em `$keywords_string` se parece com:

```
k.keyword = 'keywordl' or k.keyword = 'keyword2' or k.keyword = 'keyword3'
```

A consulta da pesquisa construída com base no código anterior seria:

```
select s.id,
       s.headline,
       10 * sum(k.weight) / $num_keywords as score
from stories s, keywords k
where s.id = k.story
and (k.keyword = 'keyword1'
  or k.keyword = 'keyword2'
  or k.keyword = 'keyword3')
group by s.id, s.headline
order by score desc, s.id desc
```

O cálculo da porcentagem é a soma dos pesos de todas as palavras-chave correspondentes dividida pelo número de palavras-chave pesquisadas e então multiplicada por dez. Isso favorece pesquisas em que todas as palavras-chave inseridas correspondam a palavras-chave no banco de dados.

Como os pesos variam de 1 a 10, o valor máximo para a contagem é 100. Uma pesquisa de três palavras-chave só teria uma correspondência de 100% com uma matéria se todas as três fossem encontradas para essa matéria e cada uma tivesse um peso de 10.

Tela de editor

A única parte do sistema que não abrangemos é a maneira como uma matéria realmente é publicada depois que foi escrita. O script publish.php faz uma matéria ao vivo. Esse código é mostrado na Listagem 28.10.

Listagem 28.10 **publish.php – Script que lista todos os documentos de modo que o editor possa escolher aqueles que são exibidos no site ao vivo**

```
<?php
  include_once('include_fns.php');

  if (!check_auth_user( ))
  {
    login_form( );
  }
  else
  {
    $handle = db_connect( );

    $writer = get_writer_record($_SESSION['auth_user']);

    echo '<p>Welcome, '.$writer['full_name'];
    echo ' (<a href="logout.php">Logout</a>) (<a href="index.php">Menu</a>)
           (<a href="../">Public Site</a>) </p>';

    $query = "select * from stories s, writer_permissions wp
              where wp.writer = '{$_SESSION['auth_user']}' and
                    s.page = wp.page
              order by modified desc";
    $result = $handle->query($query);

    echo '<h1>Editor admin</h1>';

    echo '<table>';
    echo '<tr><th>Headline</th><th>Last modified</th></tr>';
    while ($story = $result->fetch_assoc( ))
    {
      echo '<tr><td>';
      echo $story['headline'];
      echo '</td><td>';
```

Listagem 28.10 **Continuação**

```
        echo date('M d, H:i', $story['modified']);
        echo '</td><td>';
        if ($story[published])
        {
          echo '[<a href="unpublish_story.php?story='.$story['id'].
               '">unpublish</a>] ';
        }
        else
        {
          echo '[<a href="publish_story.php?story='.$story['id'].'">publish</a>] ';
          echo '[<a href="delete_story.php?story='.$story['id'].'">delete</a>] ';
        }
        echo '[<a href="story.php?story='.$story['id'].'">edit</a>] ';

        echo '</td></tr>';
      }
      echo '</table>';
    }
?>
```

Esse script deve ser disponibilizado apenas para as pessoas que têm autorização de publicar matérias no site ao vivo. Em nossa aplicação de exemplo, não há ninguém que tenha permissão de enviar matérias para a página. Em um sistema real, é provável requerer uma nível mais alto de permissão, ou pelo menos requerer que alguém, que não seja o autor original, leia e aprove o material.

O script `publish.php` é muito semelhante ao `stories.php` exceto que o editor recebe uma tela que mostra as matérias para cada escritor, não apenas as suas. A instrução `if` assegura que opções apropriadas sejam apresentadas para cada matéria. As matérias publicadas podem ser marcadas como não-publicadas e as matérias não-publicadas podem ser marcadas como publicadas ou excluídas.

Esses três links enviam para `unpublish_story.php`, `publish_story.php` e `delete_story.php`, respectivamente.

O script `publish_story.php` utiliza a seguinte consulta de SQL:

```
update stories set published = $now
       where id = $story
```

Essa consulta marcará uma matéria como publicada e a autorizará para visualização pública.

De maneira semelhante, `unpublish_story.php` utiliza a seguinte consulta para marcar uma matéria como não-publicada e interromper sua exibição para o público:

```
update stories set published = null
       where id = $story
```

O link de edição aparece independente de uma matéria ser marcada como publicada ou não, então o editor sempre pode fazer alterações. Isso é diferente para o nível de acesso dos escritores, onde eles só podem modificar uma matéria antes de sua publicação.

Estendendo o projeto

Há várias maneiras de estender esse projeto para torná-lo um sistema de gerenciamento de conteúdo mais abrangente:

- Você poderia permitir que grupos de usuários trabalhassem em matérias conjuntamente (colaboração).
- Você poderia implementar um layout de página mais flexível para que os editores pudessem posicionar texto e imagens na página.

- Você poderia construir uma biblioteca de imagens para que imagens freqüentemente utilizadas não fossem duplicadas e para que palavras-chave de pesquisa pudessem ser atribuídas a imagens também, além de serem atribuídas ao texto de matéria.
- Você também poderia adicionar a funcionalidade de verificação ortográfica para o editor de conteúdo. Uma verificação poderia ser implementada utilizando, por exemplo, as bibliotecas Aspell ou Ispell.

A seguir

No próximo projeto, construiremos uma interface baseada na Web que permite verificar e enviar e-mail a partir da Web utilizando IMAP.

29

Construindo um serviço de e-mail baseado na Web

HOJE EM DIA, COM CADA VEZ MAIS FREQÜÊNCIA, os sites querem oferecer a seus usuários serviço de e-mail baseado na Web. Este capítulo explica como implementar uma interface Web para um servidor existente de correio utilizando a biblioteca PHP IMAP. Você pode utilizá-la para verificar sua própria caixa de correio existente por meio de uma página Web ou talvez estendê-la a fim de suportar muitos usuários para um sistema de e-mail de massa baseado na Web como o Hotmail.

Neste projeto, construiremos um cliente de e-mail, Warm Mail, que permitirá aos usuários:

- Conectar-se à suas contas em servidores de correio POP3 ou IMAP;
- Ler correio;
- Enviar correio;
- Responder às mensagens de correio;
- Encaminhar mensagens de correio;
- Excluir correio de suas contas.

O problema

Para que um usuário seja capaz de ler seu correio, precisaremos encontrar uma maneira de nos conectar a seu servidor de correio. Geralmente essa não será a mesma máquina que a do servidor da Web. Precisaremos de uma maneira de interagir com sua caixa de correio, para ver que mensagens foram recebidas e lidar com cada mensagem individualmente.

Dois protocolos principais são suportados por servidores de correio para ler caixas de correio de usuário: POP3 e IMAP. Se possível, devemos suportar esses dois protocolos. O POP3 significa a versão 3 de Post Office Protocol, e IMAP significa Internet Message Access Protocol.

A principal diferença entre esses dois é que POP3 é destinado a, e normalmente utilizado por, pessoas que se conectam a uma rede por um curto período de tempo para fazer download e excluir correio de um servidor. O IMAP é destinado à utilização on-line, para interagir com correio que permanece continuamente no servidor remoto. O IMAP apresenta alguns recursos mais avançados que não utilizaremos aqui.

Se estiver interessado nas diferenças entre esses protocolos, você pode consultar suas RFCs (RFC 1939 para a versão 3 do POP e a RFC 3501 para a versão 4 rev1 do IMAP). Um excelente artigo que compara os dois protocolos pode ser encontrado em:

http://www.imap.org/papers/imap.vs.pop.brief.html

Nenhum desses protocolos é projetado para enviar correio – para isso devemos utilizar o *SMTP* (*Simple Mail Transfer Protocol*), que utilizamos antes a partir do PHP via função `mail( )`. Esse protocolo é descrito em RFC 821.

Componentes da solução

O PHP tem excelente suporte de IMAP e POP3, mas esse suporte é fornecido via biblioteca de funções do IMAP. A fim de utilizar o código apresentado neste capítulo, você precisará ter instalado a biblioteca de IMAP. Você pode determinar se já tem isso instalado examinando a saída da função `phpinfo( )`.

Se estiver utilizando o Linux ou Unix e não possui uma biblioteca IMAP instalada, você precisará fazer download das bibliotecas necessárias. Obtenha a última versão via FTP, em ftp://ftp.cac.washington.edu/imap/.

No Unix, faça download do código-fonte e compile-o para seu sistema operacional. Alguns usuários informaram dificuldades ao compilar novas versões da biblioteca IMAP com o PHP. Se você tiver dificuldades, a solução parece ser voltar a usar IMAP-2001, uma versão mais antiga, porém estável.

Em seguida, crie um diretório para os arquivos IMAP dentro de seu diretório include, chamado, digamos, `imap`. (*Não* copie apenas os arquivos para o diretório include básico porque isso pode gerar conflitos.) Dentro do novo diretório, crie dois subdiretórios chamados `imap/lib/` e `imap/include/`. Copie todos os arquivos `*.h` da instalação para `imap/include/`. Quando você realizou a compilação, um arquivo chamado `c-client.a` foi criado. Renomeie-o como `libc-client.a` e copie-o para o diretório `imap/lib/`.

Então é preciso executar o script de configuração do PHP, adicionando a diretiva `--with-imap=`*`dirname`* (onde *`dirname`* é o nome do diretório criado) para quaisquer outros parâmetros utilizados, e recompile o PHP.

Para usar a extensão IMAP com o Windows, abra o arquivo `php.ini` e retire essa linha de comentário:

```
extension=php_imap.dll
```

Em seguida, reinicie o servidor Web.

É possível confirmar se a extensão IMAP está instalada executando a função `phpinfo( )`. Deve ser exibida uma seção para IMAP.

Algo a notar é que embora essas funções sejam chamadas de funções de IMAP elas também funcionam igualmente bem com o POP3 e o NNTP (Network News Transfer Protocol). Nós os utilizaremos para IMAP e POP3, mas a aplicação Warm Mail poderia facilmente ser estendida para utilizar NNTP e ser um leitor de news bem como cliente de correio.

Existe um número muito grande de funções nessa biblioteca, mas a fim de implementar a funcionalidade dessa aplicação, utilizaremos somente algumas. Explicaremos essas funções à medida que as utilizarmos, mas esteja ciente de que há muitas outras funções. Veja a documentação se suas necessidades forem diferentes das nossas ou se quiser adicionar recursos extras à sua aplicação.

Você pode construir uma aplicação de correio relativamente útil com apenas um pequeno grupo de funções predefinidas. Isso significa que você só precisa examinar uma parte da documentação. As funções de IMAP que utilizamos neste capítulo são:

- `imap_open( )`
- `imap_close( )`
- `imap_headers( )`
- `imap_header( )`
- `imap_fetchheader( )`

- `imap_body( )`
- `imap_delete( )`
- `imap_expunge( )`

Para um usuário ler seu correio, precisaremos obter seus detalhes de servidor e de conta. Em vez de obter todos esses detalhes do usuário a cada vez, configuraremos um banco de dados de senhas e nomes de usuário para que possamos armazenar os detalhes dos usuários.

Freqüentemente as pessoas têm mais de uma conta de e-mail (uma particular e outra para o trabalho, por exemplo) e devemos permitir que eles se conectem a qualquer conta. Portanto, devemos permitir que as pessoas tenham diversos conjuntos de informações de conta no banco de dados.

Devemos permitir aos usuários ler, responder, encaminhar e excluir e-mails existentes, bem como enviar novas mensagens. Podemos fazer todas as partes de leitura utilizando o IMAP ou o POP3 e todas as partes de envio utilizando o SMTP com `mail( )`.

Vejamos como juntar tudo isso.

Visão geral da solução

O fluxo geral desse sistema baseado na Web não é muito diferente de outros clientes de e-mail. Um diagrama que mostra o fluxo e módulos do sistema é mostrado na Figura 29.1.

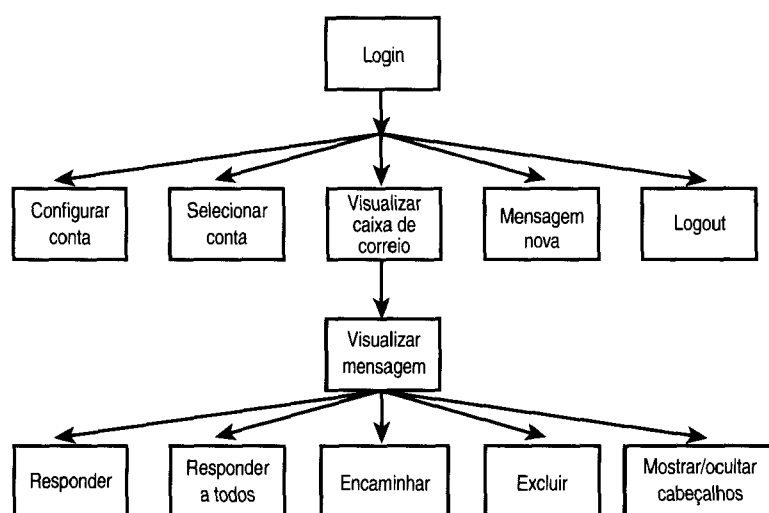



Figura 29.1 A interface para o Warm Mail fornece as funcionalidades de caixa de correio e de mensagem.

Como você pode ver, vamos primeiro requerer que o usuário efetue logon, e depois fornecer a ele uma escolha de opções. Ele será capaz de configurar uma nova conta de correio ou selecionar uma de suas contas existentes para utilização. Ele também será capaz de visualizar correio recebido – respondendo-o, encaminhando-o ou excluindo-o – e enviar novo correio.

Além disso, forneceremos a opção de visualizar cabeçalhos detalhados para uma mensagem particular. Visualizar os cabeçalhos completos pode informar bastante sobre uma mensagem. Você pode ver de qual máquina o correio veio – uma ferramenta útil para monitorar inundação. Você pode ver qual máquina o encaminhou e em que horário o correio alcançou cada host – útil para atribuir responsabilidade por mensagens atrasadas. Você também pode ser capaz de ver qual cliente de e-mail o remetente utilizou se a aplicação adicionar informações opcionais aos cabeçalhos.

Utilizamos uma arquitetura de aplicação ligeiramente diferente para este projeto. Em vez de ter um conjunto de scripts, um para cada módulo, temos um script ligeiramente mais longo, `index.php`, que funciona como o loop de evento de um programa baseado em GUI. Cada ação que realizamos

no site pressionando um botão nos levará de volta para index.php, mas com um parâmetro diferente. Dependendo do parâmetro, diferentes funções serão chamadas para mostrar a saída apropriada para o usuário. As funções estão nas bibliotecas de funções, como de costume.

Essa arquitetura é adequada para pequenas aplicações como essa. Essa arquitetura se ajusta a aplicações que são extremamente baseadas em eventos, em que as ações de usuário desencadeiam a funcionalidade. Utilizar um único handler de evento não é muito adequado para arquiteturas maiores ou projetos que estão sendo realizados por uma equipe. Um resumo dos arquivos no projeto Warm Mail é mostrado na Tabela 29.1.

Tabela 29.1 **Arquivos na aplicação Warm Mail**

Nome	Tipo	Descrição
index.php	Aplicação	O script principal que executa a aplicação inteira.
include_fns.php	Funções	Coleção de arquivos include para essa aplicação.
data_valid_fns.php	Funções	Coleção de funções para validar dados de entrada.
db_fns.php	Funções	Coleção de funções para conectar-se ao banco de dados mail.
mail_fns.php	Funções	Coleção de funções relacionadas ao e-mail para abrir caixas de correio, ler correio e assim por diante.
output_fns.php	Funções	Coleção de funções para enviar HTML para saída.
user_auth_fns.php	Funções	Coleção de funções para autenticar usuários.
create_database.sql	SQL	SQL para configurar o banco de dados book_sc e configurar um usuário.

Vamos seguir em frente e examinar a aplicação.

Configurando o banco de dados

O banco de dados para Warm Mail é relativamente simples porque não vamos realmente armazenar nenhum e-mail nele.

Precisaremos armazenar os usuários do sistema. Para cada usuário, precisaremos armazenar os seguintes campos:

- username – Nome de usuário preferido para Warm Mail;
- password – Senha preferida para Warm Mail;
- address – Endereço preferido de e-mail, que aparecerá no campo From de e-mails que eles enviam a partir do sistema;
- displayname – O nome "legível por humano" que eles gostariam de exibir no e-mail para os outros correios.

Precisaremos também armazenar cada conta que usuários gostariam de verificar com o sistema. Para cada conta, precisaremos armazenar as seguintes informações:

- username – O usuário do Warm Mail a que essa conta pertence.
- server – A máquina em que a conta reside, por exemplo, host local ou mail.tangledweb.com.au.
- port – A porta a que se conectar ao utilizar essa conta. Normalmente essa será 110 para servidores de POP3 e 143 para servidores de IMAP.

- type – O protocolo utilizado para conectar-se a esse servidor, 'POP3' ou 'IMAP'.
- remoteuser – O nome de usuário para conectar-se ao servidor de correio.
- remotepassword – A senha para conectar-se ao servidor de correio.
- accountid – Uma chave única para identificar contas.

Você pode configurar o banco de dados para essa aplicação executando a SQL mostrada na Listagem 29.1.

Listagem 29.1 **create_database.sql – SQL para criar o banco de dados de correio**

```
create database mail;

use mail;

create table users
(
  username char(16) not null primary key,
  password char(16) not null,
  address char(100) not null,
  displayname char(100) not null
);

create table accounts
(
  username char(16) not null,
  server char(100) not null,
  port int not null,
  type char(4) not null,
  remoteuser char(50) not null,
  remotepassword char(50) not null,
  accountid int unsigned not null auto_increment primary key
);

grant select, insert, update, delete
on mail.*
to mail@localhost identified by 'password';
```

Lembre-se de que você pode executar essa instrução de SQL digitando:

```
mysql -u root -p < create_database.sql
```

Você precisará fornecer sua senha de root. Você deve alterar a senha para o usuário de correio em create_database.sql e em db_fns.php antes de executá-la.

No CD-ROM, também fornecemos um arquivo de SQL chamado populate.sql. Nessa aplicação, não vamos criar um registro de usuário ou processo de administração. Você mesmo pode adicionar um se quiser utilizar esse software em uma escala maior, mas se quiser esse software para utilização pessoal, você apenas precisará inserir você próprio no banco de dados. O script populate.sql fornece uma forma de fazer isso, então insira seus detalhes nele e execute-o para se configurar como um usuário.

Arquitetura de script

Como mencionamos antes, essa aplicação utiliza um script para controlar tudo. Esse script é chamado index.php. Esse código é mostrado na Listagem 29.2. Esse script é bem longo, mas nós o examinaremos seção por seção.

Listagem 29.2 **index.php** – O backbone do sistema Warm Mail

```
<?php
// Esse arquivo é o corpo principal da aplicação Warm Mail.
// Funciona basicamente como uma máquina de estado e mostra aos usuários a
// saída para ação que escolheram.

//*****************************************************************************
// Etapa 1: pré-processamento
// Faz qualquer processamento antes que o cabaçalho da página é enviado
// e decide que detalhes devem ser exibidos nos cabeçalhos da página
//*****************************************************************************

  include ('include_fns.php');
  session_start( );
  //cria nomes de variáveis abreviados
  $username = $_POST['username'];
  $passwd = $_POST['passwd'];
  $action = $_REQUEST['action'];
  $account = $_REQUEST['account'];
  $messageid = $_GET['messageid'];

  $to =  $_POST['to'];
  $cc =  $_POST['cc'];
  $subject =  $_POST['subject'];
  $message =  $_POST['message'];

  $buttons = array( );

  //acrescenta a essa string se algo for processado antes da saída do cabaçalho
  $status = '';

  // precisa processar solicitações de logon ou logout antes de tudo
  if($username||$password)
  {
    if(login($username, $passwd))
    {
      $status .= '<p>Logged in successfully.</p><br /><br /><br /><br />
       <br /><br />';
      $_SESSION['auth_user'] = $username;
      if(number_of_accounts($_SESSION['auth_user'])==1)
      {
        $accounts = get_account_list($_SESSION['auth_user']);
        $_SESSION['selected_account'] = $accounts[0];
      }
    }
    else
    {
      $status .= '<<p>Sorry, we could not log you in with that
                  username and password.</p><br /><br /><br /><br />
                  <br /><br />';
    }
  }
  if($action == 'log-out')
  {
    session_destroy( );
    unset($action);
    $_SESSION=array( );
  }

  //precisa processar escolha, exclusão ou armazenamento da conta antes de gerar o cabaçalho
  switch ( $action )
```

Listagem 29.2 **Continuação**

```
{
  case 'delete-account' :
  {
    delete_account($_SESSION['auth_user'], $account);
    break;
  }
  case 'store-settings' :
  {
    store_account_settings($_SESSION['auth_user'], $_POST);
    break;
  }
  case 'select-account' :
  {
    // se tiver escolhido uma conta válida, armazena essa conta com uma variável de sessão
    if($account&&account_exists($_SESSION['auth_user'], $account))
    {
      $_SESSION['selected_account'] = $account;
    }
  }
}
// configura os botões que estarão na barra de ferramentas
$buttons[0] = 'view-mailbox';
$buttons[1] = 'new-message';
$buttons[2] = 'account-setup';
//oferece apenas um botão de logout se estiver conectado
if(check_auth_user( ))
{
  $buttons[4] = 'log-out';
}

//*****************************************************************************
// Etapa 2: cabaçalhos
// Envia os cabeçalhos HTML e barra de menu apropriada para ação atual
//*****************************************************************************
  if($action)
  {
    // exibe o cabeçalho com o nome da aplicação e a descrição da página ou ação
    do_html_header($_SESSION['auth_user'], "Warm Mail - ".
                   format_action($action),
                   $_SESSION['selected_account']);
  }
  else
  {
    // exibe cabaçalho apenas com o nome da aplicação
    do_html_header($_SESSION['auth_user'], "Warm Mail",
     $_SESSION['selected_account']);
  }

  display_toolbar($buttons);

//*****************************************************************************
// Etapa 3: corpo
// Dependendo da ação, mostra o conteúdo apropriado do corpo principal
//*****************************************************************************
  //exibe qualquer texto gerado por funções chamadas antes do cabaçalho
  echo $status;

  if(!check_auth_user( ))
  {
    echo '<p>You need to log in';
```

Listagem 29.2 **Continuação**

```
    if($action&&$action!='log-out')
      echo ' to go to '.format_action($action);
    echo '.</p><br /><br />';
    display_login_form($action);
  }
  else
  {
    switch ( $action )
    {
      // se escolhermos configurar uma nova conta ou acabamos de adicionar ou
      // excluimos uma conta, mostra página de configuração da conta
      case 'store-settings' :
      case 'account-setup' :
      case 'delete-account' :
      {
        display_account_setup($_SESSION['auth_user']);
        break;
      }
      case 'send-message' :
      {
        if(send_message($to, $cc, $subject, $message))
          echo '<p>Message sent.</p><br /><br /><br /><br /><br /><br />';
        else
          echo '<p>Could not send message.</p><br /><br /><br /><br />
                <br /><br />';
        break;
      }
      case 'delete' :
      {
         delete_message($_SESSION['auth_user'],
                        $_SESSION['selected_account'], $messageid);
        //note deliberadamente nenhum 'break' -- continuaremos com o próximo caso
      }
      case 'select-account' :
      case 'view-mailbox' :
      {
        // se mailbox acabou de ser escolhido, ou se view mailbox foi escolhido, exibe a caixa de correio
        display_list($_SESSION['auth_user'],
          $_SESSION['selected_account']);
        break;
      }
      case 'show-headers' :
      case 'hide-headers' :
      case 'view-message' :
      {
        // se acabamos de selecionar uma mensagem da lista ou estávamos visualizando
        // uma mensagem e escolhemos ocultar ou visualizar os cabeçalhos, carrega uma mensagem
        $fullheaders = ($action=='show-headers');
        display_message($_SESSION['auth_user'],
                        $_SESSION['selected_account'],
                        $messageid, $fullheaders);
        break;
      }
      case 'reply-all' :
      {
        //configura cc como a linha cc antiga
        if(!$imap)
          $imap = open_mailbox($_SESSION['auth_user'],
                               $_SESSION['selected_account']);
        if($imap)
```

Listagem 29.2 Continuação

```
      {
        $header = imap_header($imap, $messageid);
        if($header->reply_toaddress)
          $to = $header->reply_toaddress;
        else
          $to = $header->fromaddress;
        $cc = $header->ccaddress;
        $subject = 'Re: '.$header->subject;
        $body = add_quoting(imap_body($imap, $messageid));
        imap_close($imap);

        display_new_message_form($_SESSION['auth_user'],
        $to, $cc, $subject, $body);
      }
      break;
    }
    case 'reply' :
    {
      //configura o endereço como reply-to ou from da mensagem atual
      if(!$imap)
        $imap = open_mailbox($_SESSION['auth_user'],
                             $_SESSION['selected_account']);
      if($imap)
      {
        $header = imap_header($imap, $messageid);
        if($header->reply_toaddress)
          $to = $header->reply_toaddress;
        else
          $to = $header->fromaddress;
        $subject = 'Re: '.$header->subject;
        $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
        imap_close($imap);

        display_new_message_form($_SESSION['auth_user'],
                                 $to, $cc, $subject, $body);
      }

      break;
    }
    case 'forward' :
    {
      //define mensagem conforme corpo citado da mensagem atual
      if(!$imap)
        $imap = open_mailbox($_SESSION['auth_user'],
                             $_SESSION['selected_account']);
      if($imap)
      {
        $header = imap_header($imap, $messageid);
        $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
        $subject = 'Fwd: '.$header->subject;
        imap_close($imap);

        display_new_message_form($_SESSION['auth_user'],
                                 $to, $cc, $subject, $body);
      }
      break;
    }
    case 'new-message' :
    {
      display_new_message_form($_SESSION['auth_user'],
```

Listagem 29.2 Continuação

```
                                        $to, $cc, $subject, $body);
          break;
        }
      }
    }
//*****************************************************************************
// Etapa 4: rodapé
//*****************************************************************************
  do_html_footer( );
?>
```

Esse script utiliza uma abordagem de tratamento de evento. Ele contém o conhecimento ou a lógica sobre que função precisa ser chamada para cada evento. Os eventos nesse caso são desencadeados pelo usuário clicando nos vários botões no site, cada um dos quais seleciona uma ação. A maioria de botões é produzida pela função `display_button( )`, mas a função `display_form_button( )` é utilizada se ele for um botão submit. Essas duas funções estão em `output_fns.php`. Todas essas pulam para URLs na forma:

```
index.php?action=log-out
```

O valor da variável `$action` quando `index.php` for chamado determina qual handler de evento ativar.

As quatro seções principais para o script são:

1. Realizamos algum processamento que deve acontecer antes de enviarmos o cabeçalho de página para o navegador, como iniciar a sessão, executar qualquer pré-processamento para a ação que o usuário selecionou e decidir como serão os cabeçalhos.
2. Processamos e enviamos para o usuário os cabeçalhos apropriados e a barra de menu adequada para a ação selecionada.
3. Escolhemos qual corpo do script executar, dependendo da ação selecionada. As diferentes ações desencadeiam diferentes chamadas de função.
4. Enviamos os rodapés de página.

Se examinar rapidamente o código para o script, você verá que essas quatro seções estão marcadas com comentários.

Para entender completamente esse script, passaremos a examinar como realmente utilizar o site ação por ação.

Efetuando logon e logout

Quando um usuário carregar a página `index.php`, ele verá a saída mostrada na Figura 29.2.

Mostrar a tela de login é o comportamento padrão para a aplicação. Com nenhuma `$action` escolhida ainda e nenhum detalhe de login fornecido, executaremos as seguintes partes do código.

Na etapa de pré-processamento, primeiramente executamos o seguinte código:

```
include ('include_fns.php');
session_start( );
```

Essas linhas iniciam a sessão que será utilizada para monitorar as variáveis de sessão `$auth_user` e `$selected_account`, que examinaremos mais tarde.

Como em nossas outras aplicações, criamos nomes de variáveis abreviados. Fizemos isso em cada script relacionado a formulário desde o Capítulo 1, então isso nem precisaria ser mencionado se não fosse pela variável `action`. Dependendo de onde aparece na aplicação, talvez ela seja uma va-

riável GET ou POST. Portanto, você deve extraí-la do array $_REQUEST. É preciso fazer o mesmo com a variável accountporque geralmente ela é acessada via GET, mas ao excluir uma conta ela é acessada via POST.

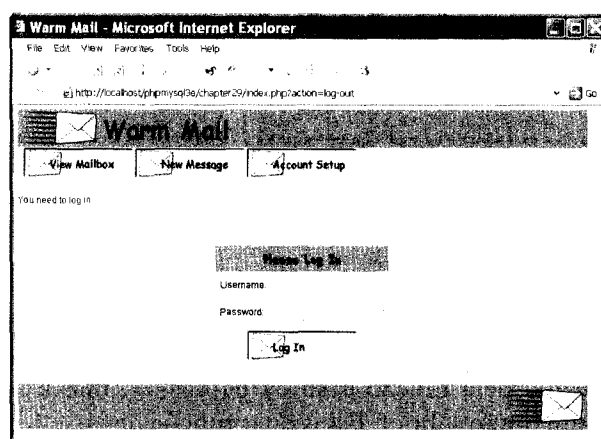



Figura 29.2 A tela de login para Warm Mail pede seu nome de usuário e senha.

Para poupar trabalho ao personalizar a interface com o usuário, os botões que aparecem na barra de ferramentas são controlados por um array. Declaramos um array vazio:

```
$buttons = array( );
```

e configuramos os botões que queremos na página:

```
$buttons[0] = 'view-mailbox';
$buttons[1] = 'new-message';
$buttons[2] = 'account-setup';
```

Para a etapa de cabeçalho, imprimimos um cabeçalho básico:

```
do_html_header($_SESSION['auth_user'], 'Warm Mail',
                              $_SESSION['selected_account']);
...
display_toolbar($buttons);
```

Esse código imprime o título e a barra de cabeçalho e então a barra de ferramentas de botões, que pode ser vista na Figura 29.2. Essas funções podem ser encontradas na biblioteca de funções output_fns.php, mas como você pode facilmente ver seu efeito na figura, não passaremos por elas aqui.

Agora chegamos ao corpo do código:

```
if(!check_auth_user( ))
{
   echo '<p>You need to log in';
   if($action&&$action!='log-out')
   echo ' to go to '.format_action($action);
   echo '.</p><br /><br />';
   display_login_form($action);
}
```

A função check_auth_user( ) é da biblioteca user_auth_fns.php. Utilizamos código muito parecido em alguns projetos anteriores – ele marca se o usuário efetuar logon. Se ele não estiver, que é o caso aqui, mostraremos a ele um formulário de login, que você pode ver na Figura 29.2. Desenhamos esse formulário na função display_login_form( ) de output_fns.php.

Se o usuário preencher o formulário corretamente e pressionar o botão Log In, verá a saída mostrada na Figura 29.3.

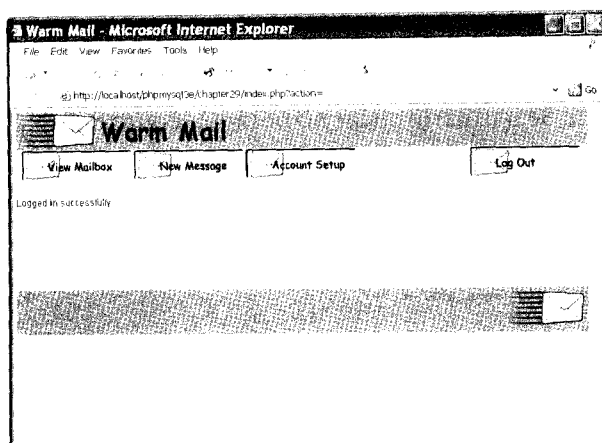



Figura 29.3 Depois de login bem-sucedido, o usuário pode começar a utilizar a aplicação.

Nessa execução do script, ativaremos diferentes seções de código. O formulário de login tem dois campos, `$username` e `$password`. Se esses dois campos forem preenchidos, o seguinte segmento de código de pré-processamento será ativado:

```
if($username||$password)
{
  if(login($username, $passwd))
  {
    $status .= '<p>Logged in successfully.</p><br /><br /><br /><br />
     <br /><br />';
    $_SESSION['auth_user'] = $username;
    if(number_of_accounts($_SESSION['auth_user'])==1)
    {
      $accounts = get_account_list($_SESSION['auth_user']);
      $_SESSION['selected_account'] = $accounts[0];
    }
  }
  else
  {
    $status .= '<p>Sorry, we could not log you in with that
                username and password.</p><br /><br /><br /><br />
                <br /><br />';
  }
}
```

Como você pode ver, o código chama a função `login( )`, que é semelhante à utilizada nos Capítulos 26 e 27. Se tudo vai bem, registramos o nome de usuário na variável de sessão `$auth_user`.

Além de configurar os botões que vimos enquanto não estávamos conectados, adicionamos outro botão para permitir ao usuário efetuar logout novamente, assim:

```
if(check_auth_user( ))
{
    $buttons[4] = 'log-out';
}
```

Você pode ver esse botão Log Out na Figura 29.3.

Na etapa de cabeçalho, novamente exibimos o cabeçalho e os botões. No corpo, exibimos a mensagem de status que configuramos anteriormente:

```
echo $status;
```

Depois disso, é só uma questão de imprimir o rodapé e esperar para ver o que o usuário fará em seguida.

Configurando contas

Quando um usuário começa a utilizar o sistema Warm Mail pela primeira vez, ele precisa configurar algumas contas de e-mail. Se o usuário clicar no botão Account Setup, isso configurará a variável `action` como `account-setup` e chamará novamente o script `index.php`. O usuário então verá a saída mostrada na Figura 29.4.

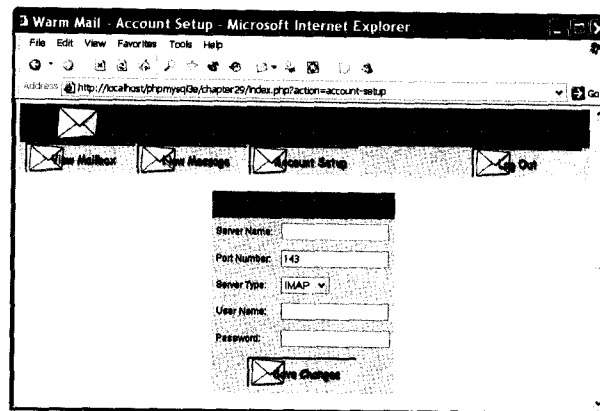



Figura 29.4 Um usuário precisa configurar seus detalhes de conta de e-mail antes de poder ler seu e-mail.

Examine outra vez o script na Listagem 29.2. Dessa vez, por causa do valor de `$action`, obtemos um comportamento diferente. Obtemos um cabeçalho ligeiramente diferente:

```
do_html_header($_SESSION['auth_user'], 'Warm Mail - '.
             format_action($action), $_SESSION['selected_account']);
```

Sobretudo, obtemos um corpo diferente:

```
case 'store-settings' :
case 'account-setup' :
case 'delete-account' :
{
   display_account_setup($HTTP_SESSION_VARS['auth_user']);
   break;
}
```

Esse é o padrão típico: cada comando chama uma função. Nesse caso, chamamos a função `display_account_setup( )`. O código para essa função é mostrado na Listagem 29.3.

Listagem 29.3 **Função `display_account_setup( )` de `output_fns.php` – Função para obter e exibir detalhes de conta**

```
function display_account_setup($auth_user)
{
   //exibe formulário de 'new account' vazio

   display_account_form($auth_user);
   $list = get_accounts($auth_user);
   $accounts = sizeof($list);

   // exibe cada conta armazenada
   foreach($list as $key => $account)
   {
     // exibe formulário para detalhes de cada conta.
     // observe que enviaremos a senha para todas as contas na HTML
```

Listagem 29.3 **Continuação**

```
    // essa não é realmente uma idéia muito boa
    display_account_form($auth_user, $account['accountid'], $account['server'],
                        $account['remoteuser'], $account['remotepassword'],
                        $account['type'], $account['port']);
  }
}
```

Quando chamamos essa função, ela exibe um formulário em branco para adicionar uma nova conta, seguida por formulários editáveis contendo cada uma das contas atuais de e-mail do usuário. A função `display_account_form( )` exibirá o formulário que podemos ver na Figura 29.4. Você pode ver que utilizamos essa função de duas maneiras diferentes aqui: sem parâmetros para exibir um formulário vazio e com um conjunto completo de parâmetros para exibir um registro existente. Essa função está na biblioteca `output_fns.php`; ela simplesmente envia a HTML para saída; então, não nos deteremos com ela aqui.

A função que recupera qualquer conta existente é `get_accounts( )`, a partir da biblioteca `mail_fns.php`. Essa função é mostrada na Listagem 29.4.

Listagem 29.4 **Função `get_accounts( )` de `mail_fns.php` – Função para recuperar todos os detalhes de conta para um usuário particular**

```
function get_accounts($auth_user)
{
  $list = array( );
  if($conndb_connect( ))
  {
    $query = "select * from accounts where username = '$auth_user'";
    $result = $conn_query($query);
        if($result)
    {
      while($settings = $result->fetch_assoc( ))
        array_push( $list, $settings);
    }
    else
      return false;
  }
  return $list;
}
```

Como você pode ver, essa função conecta-se ao banco de dados, recupera todas as contas para um usuário particular e as retorna como um array.

Criando uma nova conta

Se um usuário preencher o formulário de conta e clicar no botão Save Changes, a ação `store-settings` será ativada. Vejamos o código de tratamento de eventos para isso a partir de `index.php`. Na etapa de pré-processamento, executamos o seguinte código:

```
case 'store-settings' :
{
  store_account_settings($_SESSION['auth_user'], $_POST);
  break;
}
```

A função `store_account_settings( )` grava os novos detalhes de conta no banco de dados. O código para essa função é mostrado na Listagem 29.5.

Listagem 29.5 **Função `store_account_settings( )` de `mail_fns.php` – Função para salvar novos detalhes de conta para um usuário**

```
function store_account_settings($auth_user, $settings)
{
  if(!filled_out($settings))
  {
    echo 'All fields must be filled in.  Try again.<br /><br />';
    return false;
  }
  else
  {
    if($settings['account']>0)
      $query = "update accounts  set server = '$settings[server]',
                  port = $settings[port], type = '$settings[type]',
                  remoteuser = '$settings[remoteuser]',
                  remotepassword = '$settings[remotepassword]'
                where accountid = $settings[account]
                  and username = '$auth_user'";
    else
      $query = "insert into accounts values ('$auth_user',
                     '$settings[server]', $settings[port],
                     '$settings[type]', '$settings[remoteuser]',
                     '$settings[remotepassword]', NULL)";
    if($conn=db_connect( ))
   {
     $result=$conn->query($query);
     if ($result)
       return true;
     else
       return false;
   }
   else
   {
     echo 'could not store changes.<br /><<br /><br /><br /><br />';
     return false;
   }
  }
}
```

Como você pode ver, duas escolhas dentro dessa função correspondem a inserir uma nova conta ou atualizar uma conta existente. A função executa a consulta apropriada para salvar os detalhes da conta.

Depois de armazenar os detalhes de conta, voltamos para `index.php`, para a etapa de corpo principal:

```
case 'store-settings' :
case 'account-setup' :
case 'delete-account' :
{
  display_account_setup($_SESSION['auth_user']);
 break;
}
```

Como você pode ver, executamos a função `display_account_setup( )` da maneira anterior a fim de listar os detalhes de conta de usuário. A conta recém-adicionada será agora incluída.

Modificando uma conta existente

O processo de modificar uma conta existente é muito semelhante. O usuário pode alterar os detalhes de conta e clicar no botão Save Changes. Novamente, isso desencadeará a ação store-settings, mas dessa vez atualizará os detalhes de conta em vez de inseri-los.

Excluindo uma conta

Para excluir uma conta, o usuário pode clicar no botão Delete Account que é mostrado sob cada listagem de conta. Isso ativa a ação delete-account.

Na seção de pré-processamento do script index.php, executaremos o seguinte código:

```
case 'delete-account' :
{
   delete_account($_SESSION['auth_user'], $account);
   break;
}
```

Esse código chama a função delete_account( ). O código para essa função é mostrado na Listagem 29.6. A exclusão de contas precisa ser tratada antes do cabeçalho porque uma escolha de qual conta utilizar está dentro do cabeçalho. A lista de contas precisa ser atualizada antes que isso possa ser corretamente obtido.

Listagem 29.6 **Função delete_account( ) de mail_fns.php – Função para excluir os detalhes de uma única conta**

```
function delete_account($auth_user, $accountid)
{
   //exclui do DB uma conta deste usuário

   $query = "delete from accounts where accountid='$accountid'
                                    and username ='$auth_user'";
  if($conn=db_connect( ))
   {
    $result = $conn_query($query);
   }
   return $result;
}
```

Depois que a execução retornar para index.php, a etapa do corpo executará o seguinte código:

```
case 'store-settings' :
case 'account-setup' :
case 'delete-account' :
{
   display_account_setup($_SESSION['auth_user']);
   break;
}
```

Você reconhecerá esse como o mesmo código que executamos anteriormente – ele simplesmente exibe a lista de contas do usuário.

Lendo correio

Depois que o usuário configurou algumas contas, ele pode passar para a função principal: conectar-se a essas contas e ler correio.

Selecionando uma conta

Precisamos selecionar uma das contas de usuário para ler o correio. A conta atualmente selecionada está armazenada na variável de sessão $selected_account.

Se o usuário tiver uma única conta registrada no sistema, ela será automaticamente selecionada quando ele efetuar logon, assim:

```
if(number_of_accounts($_SESSION['auth_user'])==1)
{
  $accounts = get_account_list($_SESSION['auth_user']);
  $_SESSION['selected_account'] = $accounts[0];
}
```

A função number_of_accounts( ), de mail_fns.php, funciona se o usuário possui mais de uma conta; ela é mostrada na Listagem 29.7. A função get_account_list( ) recupera um array dos IDs de conta do usuário. Nesse caso, há exatamente um; portanto, você pode acessá-lo como o valor 0 do array.

Listagem 29.7 Função number_of_accounts( ) de mail_fns.php – Função para descobrir quantas contas um usuário registrou

```
function number_of_accounts($auth_user)
{
   // obtém o número de contas que pertencem a esse usuário

   $query = "select count(*) from accounts where username = '$auth_user'";

   if($conn=db_connect( ))
   {
     $result = $conn->query($query);
       if($result)
   }
   $row = $result->fetch-array( );
   return $row[0];
   }
{
   return 0;
}
```

A função get_account_list( ) é semelhante à função get_accounts( ) que examinamos antes, exceto que ela só recupera os nomes de conta.

Se um usuário tiver várias contas registradas, ele precisará selecionar uma para utilizar. Nesse caso, os cabeçalhos conterão uma SELECT que lista as caixas de correio disponíveis. Escolher a apropriada exibirá automaticamente a caixa de correio para essa conta. Você pode ver isso na Figura 29.5.

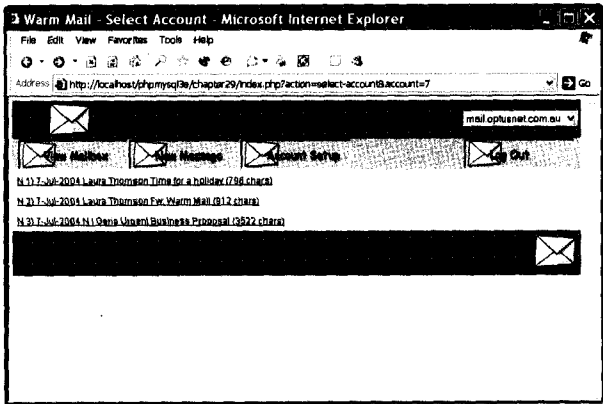



Figura 29.5 Depois que a conta de host local é selecionada a partir da caixa SELECT, o correio dessa conta é carregado e exibido.

Essa opção SELECT é gerada na função do_html_header( ) de output_fns.php, como mostra o seguinte fragmento de código:

```
// inclui a caixa de selecionar conta somente se o usuário tiver mais de uma conta
if(number_of_accounts($auth_user)>1)
{
   echo '<form target="index.php?action=open-mailbox" method="post">';
   echo '<td bgcolor="#ff6600" align="right" valign="middle">';
   display_account_select($auth_user, $selected_account);
   echo '</td>';
   echo '</form>';
}
```

Geralmente evitamos discutir a HTML utilizada nos exemplos deste livro, mas a HTML gerada pela função `display_account_select( )` merece um comentário.

Dependendo das contas que o usuário atual tem, `display_account_select( )` gerará HTML assim:

```
<select onchange="window.location=this.options[selectedIndex].value"
        name="account">
   <option value="0" selected>
     Choose Account</a>
   <option value="index.php?action=select-account&account=10">
     mail.domain.com
   </option>
   <option value="index.php?action=select-account&account=11">
     mail.server.com
   </option>
   <option value="index.php?action=select-account&account=9">
     localhost
   </option>
</select>
```

A maior parte desse código é simplesmente um elemento de seleção de HTML, mas também inclui um pouco de JavaScript. Da mesma maneira como o PHP pode gerar HTML, ele também pode ser utilizado para gerar scripts do lado cliente.

Sempre que um evento change acontecer para esse elemento, o JavaScript configurará `window.location` como o valor da opção. Se seu usuário selecionar a primeira opção na seleção, `window.location` será configurada como `'index.php?action=select-account&account=10'`. Isso resultará no carregamento desse URL. Obviamente, se o usuário tiver um navegador que não suporta o JavaScript ou tiver o JavaScript desativado, esse código não terá nenhum efeito.

A função `display_account_select( )`, de `output_fns.php`, é utilizada para obter a lista de contas disponíveis e exibir `SELECT`. Essa função também utiliza a função `get_account_list( )` que discutimos anteriormente.

Escolher uma das opções em `SELECT` ativa o evento `select_account`. Se vir o URL na Figura 29.5, você pode ver isso acrescentado ao fim do URL, junto com o ID da conta escolhida.

Isso tem dois efeitos. Primeiro, na etapa de pré-processamento de `index.php`, a conta escolhida será armazenada na variável de sessão `$selected_account`, da seguinte maneira:

```
case 'select-account' :
{
   // se tiver escolhido uma conta válida, armazena essa conta como uma variável de sessão
   if($account&&account_exists($_SESSION['auth_user'], $account))
   {
     $_SESSION['selected_account'] = $account;
   }
}
```

Segundo, quando a etapa do corpo do script for executada, o seguinte código será executado:

```
case 'select-account' :
case 'view-mailbox' :
{
```

```
    // se a caixa de correio acabou de ser escolhida, ou se view mailbox foi escolhido,
    // mostra a caixa de correio
    display_list($_SESSION['auth_user'],
                 $_SESSION['selected_account']);
    break;
}
```

Como você pode ver, tomamos a mesma ação aqui como se o usuário tivesse escolhido a opção View Mailbox. Veremos isso em seguida.

Visualizando conteúdo da caixa de correio

O conteúdo da caixa de correio pode ser visualizado com a função `display_list( )`. Isso exibe uma lista de todas as mensagens na caixa de correio. O código para essa função é mostrado na Listagem 29.8.

Listagem 29.8 **Função `display_list( )` de `output_fns.php` – Função para exibir todas as mensagens da caixa de correio**

```
function display_list($auth_user, $accountid)
{
  // mostra a lista de mensagens nessa caixa de correio

  global $table_width;

  if(!$accountid)
  {
    echo 'No mailbox selected<br /><br /><br /><br /><br /><br />.';
  }
  else
  {

    $imap = open_mailbox($auth_user, $accountid);

    if($imap)
    {
      echo "<table width=$table_width' cellspacing='0'
                 cellpadding='6'  border='0'>";

      $headers = imap_headers($imap);
      // poderíamos reformatar esses dados ou obter outros detalhes utilizando
      // imap_fetchheaders, mas isso não é um mau resumo então simplesmente ecoamos
      // cada um

      $messages = sizeof($headers);
      for($i = 0; $i<$messages; $i++)
      {
        echo '<tr><td bgcolor = "';
        if($i%2)
          echo '#ffffff';
        else
          echo '#ffffcc';
        echo '"><a href="index.php?action=view-message&messageid='.($i+1).'">';
        echo $headers[$i];
        echo "</a></td></tr>\n";
      }
      echo '</table>';
    }
    else
    {
      $account = get_account_settings($auth_user, $accountid);
```

Listagem 29.8 **Continuação**

```
        echo 'could not open mail box '.$account['server'].
             '.<br /><br /><br /><<br />';
      }
    }
}
```

Nessa função, realmente começamos a utilizar as funções de IMAP do PHP. As duas partes-chave dessa função são abrir a caixa de correio e ler os cabeçalhos da mensagem.

Abrimos a caixa de correio para uma conta de usuário com uma chamada à função `open_mailbox( )` que escrevemos em `mail_fns.php`. Essa função é mostrada na Listagem 29.9.

Listagem 29.9 **Função `open_mailbox( )` de `mail_fns.php` – Essa função conecta-se a uma caixa de correio de usuário**

```
function open_mailbox($auth_user, $accountid)
{

  // seleciona caixa de correio se houver apenas uma
  if(number_of_accounts($auth_user)==1)
  {
    $accounts = get_account_list($auth_user);
    $_SESSION['selected_account'] = $accounts[0];
    $accountid = $accounts[0];
  }

  // conecta ao sevidor POP3 ou IMAP que o usuário selecionou
  $settings = get_account_settings($auth_user, $accountid);
  if(!sizeof($settings)) return 0;
  $mailbox = '{'.$settings[server];
  if($settings[type]=='POP3')
    $mailbox .= '/pop3';

  $mailbox .= ':'.$settings[port].'}INBOX';

  // suprime o aviso, lembra de verificar o valor de retorno
@ $imap = imap_open($mailbox, $settings['remoteuser'],
                    $settings['remotepassword']);

  return  $imap;
}
```

Na realidade, abrimos a caixa de correio com a função `imap_open( )`. Essa função tem o seguinte protótipo:

```
int imap_open (string caixaDeCorreio, string nomeDoUsuário, string senha [, int opções])
```

Os parâmetros que você precisa passar para ela são:

- *`caixaDeCorreio`* – Essa string deve conter o nome de servidor e o nome da caixa de correio e, opcionalmente, um número da porta e protocolo. O formato dessa string é:

  `{nomeDoHost/protocolo:porta}nomeDaCaixa`

  Se o protocolo não for especificado, o padrão IMAP é assumido. No código que escrevemos, você pode ver que especificamos POP3 se o usuário especificou esse protocolo para uma conta particular.

Por exemplo, para ler correio da máquina local utilizando as portas padrão, utilizaríamos o seguinte nome de caixa de correio para IMAP:

```
{localhost:143}INBOX
```

e esse para o POP3

```
{localhost/pop3:110}INBOX
```

- *nomeDoUsuário* – O nome de usuário para a conta.
- *senha* – A senha para a conta.

Você também pode passar flags opcionais para especificar opções como `"open mailbox in read-only mode"`.

Algo a notar é que construímos a string de caixa de correio parte por parte com o operador de concatenação antes de passá-la para `imap_open( )`. Você precisa ter cuidado com a maneira como constrói essa string porque strings contendo `{$` causam problemas no PHP.

Essa chamada de função retorna um fluxo de IMAP se a caixa de correio puder ser aberta, e `false` se não puder.

Quando tiver concluído um fluxo de IMAP, você pode fechá-lo utilizando `imap_close(`*fluxo_de_imap*`)`. Em nossa função, o fluxo de IMAP é passado de volta para o programa principal. Portanto, utilizamos a função `imap_headers( )` para obter os cabeçalhos de e-mail para exibição:

```
$headers = imap_headers($imap);
```

Essa função retorna informações de cabeçalho para todas as mensagens de correio na caixa de correio a que nos conectamos. As informações são retornadas como um array, uma linha por mensagem. Não formatamos isso. O código envia para a saída uma linha por mensagem, então você pode ver qual é a aparência da saída examinando a Figura 29.5.

Você pode obter informações adicionais sobre cabeçalhos de e-mail utilizando a confusa função chamada de maneira semelhante `imap_header( )`. Nesse caso, a função `imap_headers( )` fornece detalhes suficientes para nossos propósitos.

Lendo uma mensagem de correio

Configuramos cada uma das mensagens na função `display_list( )` anterior para vincular-se a mensagens específicas de e-mail. Cada link está na forma:

```
index.php?action=view-message&messageid=6
```

O `messageid` é o número de seqüência utilizado nos cabeçalhos que recuperamos anteriormente. Observe que mensagens de IMAP são numeradas a partir do 1, não do 0.

Se o usuário clicar em um desses links, verá saída como aquela mostrada na Figura 29.6.

Quando inserimos esses parâmetros no script `index.php`, executamos o seguinte código:

```
case 'show-headers' :
case 'hide-headers' :
case 'view-message' :
{
  // se acabamos de selecionar uma mensagem da lista, ou estávamos visualizando
  // uma mensagem e escolhemos ocultar ou visualizar cabaçalhos, carrega uma mensagem
  $fullheaders = ($action=='show-headers');
  display_message($_SESSION['auth_user'],
                  $_SESSION['selected_account'],
                  $messageid, $fullheaders);
  break;
}
```

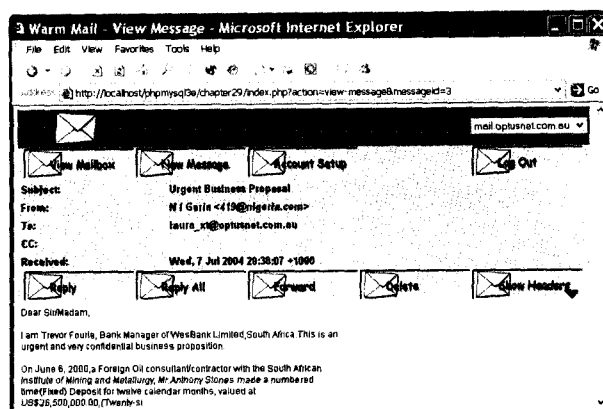



Figura 29.6 Utilizar a ação `view-message` mostra uma mensagem particular – nesse caso, é uma parte da inundação.

Você notará que estamos verificando se o valor de `$action` é igual a `'show-headers'`. Nesse caso, será `false` e `$fullheaders` será configurado como igual a `false`. Veremos a ação `'show-headers'` em breve.

A linha:

```
$fullheaders = ($action=='show-headers');
```

poderia ter sido escrita de maneira mais prolixa – e talvez mais claramente – como:

```
if($action=='show-headers')
   $fullheaders = true;
else
   $fullheaders = false;
```

Em seguida, chamamos a função `display_message( )`. A maior parte dessa função gera HTML simples, então não a examinaremos em detalhes aqui. Ela chama a função `retrieve_message( )` para obter a mensagem apropriada da caixa de correio:

```
$message = retrieve_message($auth_user, $accountid, $messageid, $fullheaders);
```

A função `retrieve_message( )` está na biblioteca `mail_fns.php`. Você pode ver o código para ela na Listagem 29.10.

Listagem 29.10 **Função `retrieve_message( )` de `mail_fns.php` – Essa função recupera uma mensagem específica de uma caixa de correio**

```
function retrieve_message($auth_user, $accountid, $messageid, $fullheaders)
{
   $message = array( );

   if(!($auth_user && $messageid && $accountid))
     return false;

   $imap = open_mailbox($auth_user, $accountid);
   if(!$imap)
     return false;

   $header = imap_header($imap, $messageid);

   if(!$header)
     return false;

   $message['body'] = imap_body($imap, $messageid);
```

Listagem 29.10 **Continuação**

```
  if(!$message['body'])
    $message['body'] = "[This message has no body]\n\n\n\n\n\n";

  if($fullheaders)
   $message['fullheaders'] = imap_fetchheader($imap, $messageid);
  else
    $message['fullheaders'] = '';

  $message['subject'] = $header->subject;
  $message['fromaddress'] =  $header->fromaddress;
  $message['toaddress'] =  $header->toaddress;
  $message['ccaddress'] =  $header->ccaddress;
  $message['date'] =  $header->date;

  // observe que podemos obter informações mais detalhadas utilizando from e to
  // em vez de fromaddress e toaddress, mas aqueles são mais fáceis

  imap_close($imap);
  return $message;
}
```

Novamente utilizamos `open_mailbox( )` para abrir a caixa de correio do usuário. Dessa vez, porém, temos em vista uma mensagem específica. Utilizando essa biblioteca de funções, fazemos download dos cabeçalhos da mensagem e do corpo da mensagem separadamente.

As três funções de IMAP que utilizamos aqui são `imap_header( )`, `imap_fetchheader( )` e `imap_body( )`. Observe que as duas funções de cabeçalho são distintas de `imap_headers( )`, que utilizamos anteriormente. Elas são chamadas de modo um pouco confuso. Em resumo:

- `imap_headers( )` – Retorna um resumo dos cabeçalhos para todas as mensagens em uma caixa de correio. Essa função os retorna como um array com um elemento por mensagem.
- `imap_header( )` – Retorna os cabeçalhos para uma mensagem específica na forma de um objeto.
- `imap_fetchheader( )` – Retorna os cabeçalhos para uma mensagem específica na forma de uma string.

Nesse caso, utilizamos `imap_header( )` para preencher campos específicos de cabeçalho e `imap_fetchheader( )` para mostrar ao usuário os cabeçalhos completos se solicitado. (Voltaremos a isso.)

Utilizamos `imap_header( )` e `imap_body( )` para construir um array contendo todos os elementos de uma mensagem em que estamos interessados. Chamamos `imap_header( )` da seguinte maneira:

```
$header = imap_header($imap, $messageid);
```

Então, podemos extrair cada um dos campos que solicitamos a partir do objeto:

```
$message['subject'] = $header->subject;
```

Chamamos `imap_body( )` para adicionar o corpo de mensagem ao nosso array da seguinte maneira:

```
$message['body'] = imap_body($imap, $messageid);
```

Por fim, fechamos a caixa de correio com `imap_close( )` e retornamos o array que construímos. A função `display_message( )` pode então exibir os campos da mensagem na forma que você pode ver na Figura 29.6.

Visualizando cabeçalhos de mensagem

Como você pode ver na Figura 29.6, há um botão Show Headers. Esse botão ativa a opção show-headers, que adiciona os cabeçalhos completos da mensagem à tela da mensagem. Se o usuário clicar nesse botão, verá saída semelhante à saída mostrada na Figura 29.7.

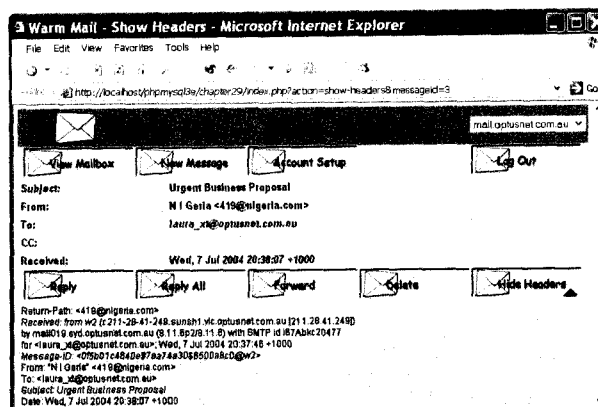



Figura 29.7 Utilizar show-headers para ver os cabeçalhos completos para essa mensagem ajudará um usuário a monitorar a fonte de um spam (inundação, correio não-solicitado).

Como você provavelmente notou, o tratamento de evento para view-message também abrange show-headers (e sua contraparte hide-headers). Se essa opção estiver selecionada, fazemos o mesmo que antes. Mas em retrieve_message( ), também nos apoderamos do texto completo dos cabeçalhos, da seguinte maneira:

```
if($fullheaders)
    $message['fullheaders'] = imap_fetchheader($imap, $messageid);
```

Então, podemos exibir esses cabeçalhos para o usuário.

Excluindo correio

Se um usuário clicar no botão Delete em um e-mail particular, ele ativará a ação "delete". Isso executará o seguinte código de index.php:

```
case 'delete' :
{
   delete_message($_SESSION['auth_user'],
                  $_SESSION['selected_account'], $messageid);
   //note deliberadamente nenhum 'break' -- continuaremos para o próximo caso
}
case 'select-account' :
case 'view-mailbox' :
{
  // se mailbox acabou de ser escolhido, ou se view mailbox foi escolhido, mostra a caixa de correio
  display_list($_SESSION['auth_user'],
    $_SESSION['selected_account']);
  break;
}
```

Como você pode ver, a mensagem é excluída utilizando a função delete_message( ) e então a caixa de correio resultante é exibida como discutido anteriormente. O código para a função delete_message( ) é mostrado na Listagem 29.11.

Listagem 29.11 **Função `delete_message( )` de `mail_fns.php` – Essa função exclui uma mensagem específica de uma caixa de correio**

```
function delete_message($auth_user, $accountid, $message_id)
{
  // exclui uma mensagem individual do servidor

  $imap = open_mailbox($auth_user, $accountid);
  if($imap)
  {
    imap_delete($imap, $message_id);
    imap_expunge($imap);
    imap_close($imap);
    return true;
  }
  return false;
}
```

Como você pode ver, essa função utiliza várias funções de IMAP. As novas são `imap_delete( )` e `imap_expunge( )`. Note que `imap_delete( )` só marca mensagens para exclusão. Você pode marcar quantas mensagens quiser. A chamada a `imap_expunge( )` realmente exclui as mensagens.

Enviando correio

Por fim chegamos ao envio de correio. Há algumas maneiras de fazer isso a partir desse script: o usuário pode enviar uma nova mensagem, responder ou encaminhar correio. Vejamos como isso funciona.

Enviando uma nova mensagem

O usuário pode escolher essa opção clicando no botão New Message. Isso ativa a ação `'new-message'`, que executa o seguinte código em `index.php`:

```
case 'new-message' :
{
  display_new_message_form($_SESSION['auth_user'],
                           $to, $cc, $subject, $body);
  break;
}
```

O formulário de nova mensagem é apenas um formulário de enviar correio. Você pode ver como ele é na Figura 29.8. Essa figura mostra na realidade um encaminhamento de correio em vez de um novo correio, mas o formulário é o mesmo. Examinaremos o encaminhamento e as respostas em seguida.

Clicar no botão Send Message invoca a ação `'send-message'`, que executa o seguinte código:

```
case 'send-message' :
{
   if(send_message($to, $cc, $subject, $message))
     echo "<p>Message sent.</p><br /><br /><br /><br /><br /><br />";
   else
     echo "<p>Could not send message.</p><br /><br /><br /><br /><br /><br />";
   break;
}
```

Esse código chama a função `send_message( )`, que realmente envia o correio. Essa função é mostrada na Listagem 29.12.

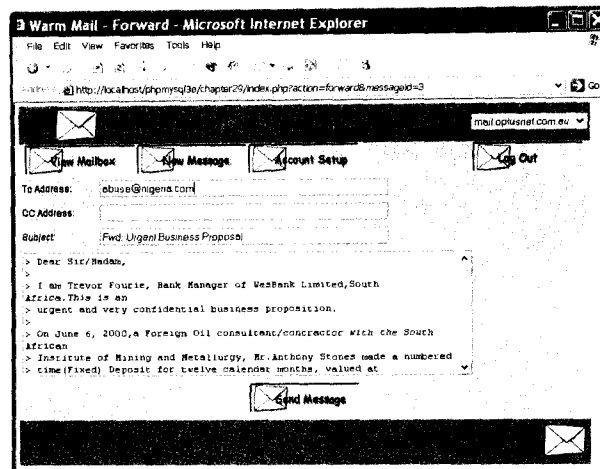



Figura 29.8 Utilizando o encaminhamento de correio, podemos informar ao usuário sobre a inundação.

Listagem 29.12 **Função send_message( ) de mail_fns.php – Essa função envia a mensagem que o usuário digitou**

```
function send_message($to, $cc, $subject, $message)
{
  // envia um e-mail via PHP

  if (!$conn=db_connect( ))
  {
    return false;
  }
  $query = 'select address from users where '
           .'username=\''.$_SESSION['auth_user']."'";

 $result = $conn->query($query);
 if (!$result)
 {
   return false;
 }
 else if ($result->num_rows==0)
 {
   return false;
 }
 else
 {
   $row = $result->fetch_object( );
   $other = 'From: '.$row->address;
   if (!empty($cc))
      $other.="\r\nCc: $cc";
   if (mail($to, $subject, $message, $other))
   {
     return true;
   }
   else
   {
     return false;
   }
 }
}
```

Como você pode ver, essa função utiliza `mail( )` para enviar o e-mail. Primeiro, porém, carrega o endereço de e-mail do usuário fora do banco de dados utilizando o campo From do e-mail.

Respondendo ou encaminhando correio

As funções Reply, Reply All e Forward enviam correio da mesma maneira que New Message. A diferença na maneira como elas funcionam é que essas funções preenchem partes do novo formulário de mensagem antes de mostrar isso ao usuário. Olhe de novo a Figura 29.8. A mensagem que estamos encaminhando foi recuada com o símbolo > e a linha Subject prefaciada com To. De maneira semelhante, as opções Reply e Reply All preencherão os destinatários, a linha de assunto e a mensagem.

O código para fazer isso é ativado na seção de corpo de `index.php`, da seguinte maneira:

```
case 'reply-all' :
{
//define cc como a linha antiga cc
  if(!$imap)
    $imap = open_mailbox($_SESSION['auth_user'],
                         $_SESSION['selected_account']);
  if($imap)
  {
    $header = imap_header($imap, $messageid);
    if($header->reply_toaddress)
      $to = $header->reply_toaddress;
    else
      $to = $header->fromaddress;
    $cc = $header->ccaddress;
    $subject = 'Re: '.$header->subject;
    $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
    imap_close($imap);

    display_new_message_form($_SESSION['auth_user'],
    $to, $cc, $subject, $body);
  }
  break;
}
case 'reply' :
{
  //configura o endereço como o reply-to ou from da mensagem atual
  if(!$imap)
    $imap = open_mailbox($_SESSION['auth_user'],
                         $_SESSION['selected_account']);
  if($imap)
  {
    $header = imap_header($imap, $messageid);
    if($header->reply_toaddress)
      $to = $header->reply_toaddress;
    else
      $to = $header->fromaddress;
    $subject = 'Re: '.$header->subject;
    $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
    imap_close($imap);

    display_new_message_form($_SESSION['auth_user'],
                             $to, $cc, $subject, $body);
  }

  break;
}
case 'forward' :
{
```

```
//configura mensagem como corpo citado da mensagem atual
if(!$imap)
  $imap = open_mailbox($_SESSION['auth_user'],
                       $_SESSION['selected_account']);
if($imap)
{
  $header = imap_header($imap, $messageid);
  $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
  $subject = 'Fwd: '.$header->subject;
  imap_close($imap);

  display_new_message_form($_SESSION['auth_user'],
                           $to, $cc, $subject, $body);
}
break;
}
```

Você pode ver que cada uma dessas opções configura os cabeçalhos apropriados, aplica formatação conforme necessário e chama a função `display_new_message_form( )` para configurar o formulário.

Esse é o conjunto completo da funcionalidade para nosso leitor de correio Web.

Estendendo o projeto

Há muitas extensões ou aprimoramentos que você poderia fazer nesse projeto. Você pode examinar o leitor de correio que utiliza normalmente para ter uma inspiração, mas algumas adições úteis são:

- Adicionar a capacidade de os usuários se registrarem nesse site. (Você poderia reutilizar algum código do Capítulo 26, para esse propósito.)
- Adicionar a capacidade de os usuários terem muitos endereços. Muitos usuários têm mais de um endereço de e-mail; talvez um endereço pessoal e um endereço comercial. Mudando seu endereço armazenado de e-mail da tabela de usuários para a tabela de contas, você poderia permitir que eles utilizassem muitos endereços. Você também precisaria alterar uma quantidade limitada das outras partes do código. O formulário send mail precisaria de uma caixa drop-down para selecionar o endereço a utilizar.
- Adicionar a capacidade de enviar, receber e visualizar correio com anexos. Se os usuários forem capazes de enviar anexos, você precisará construir capacidades de upload de arquivo conforme discutido no Capítulo 18. O envio de correio com anexos é abordado no Capítulo 30.
- Adicionar capacidades de catálogo de endereços.
- Adicionar capacidades de leitura de notícias. Ler de um servidor de NNTP utilizando as funções IMAP é quase idêntico a ler de uma caixa de correio. Você só precisa especificar um número de porta diferente e o protocolo na chamada `imap_open( )`. Em vez de nomear uma caixa de correio como INBOX, você nomeia um newsgroup a partir do qual ler. Você poderia combinar isso com as capacidades de construção de thread a partir do projeto no Capítulo 31, para construir um leitor de notícias baseado na Web com thread.

A seguir

No próximo capítulo, construiremos outro projeto relacionado a e-mail – agora, uma aplicação para suportar o envio de boletins em vários tópicos para pessoas que se inscrevem em nosso site.

30

Construindo um gerenciador de listas de mala-direta

DEPOIS QUE VOCÊ ESTABELECEU UMA BASE de assinantes para seu Web site, é interessante poder manter contato com eles enviando um boletim. Neste capítulo, implementaremos um front-end para um gerenciador de listas de mala-direta (ou MLM). Alguns MLMs permitem que cada assinante envie mensagens para outros assinantes. Nosso programa será um sistema de boletim, em que somente o administrador de lista pode enviar mensagens. Chamaremos nosso sistema de Pyramid-MLM.

Esse sistema será semelhante a outros sistemas já no mercado. Para ter alguma idéia do que estamos querendo dizer, dê uma olhada em http://www.topica.com

Nossa aplicação permitirá a um administrador criar diversas listas de mala-direta e enviar boletins para cada uma dessas listas separadamente. Essa aplicação utilizará upload de arquivo para permitir que o administrador faça upload de versões de texto e HTML de boletins criados off-line. Isso significa que os administradores podem utilizar o software que eles preferirem para criar boletins.

Os usuários serão capazes de assinar qualquer das listas no nosso site e selecionar se receberão boletins em texto ou HTML.

Discutiremos os seguintes tópicos:

- Upload de diversos arquivos;
- Anexos de e-mail codificados em mime;
- E-mail formatado em HTML;
- Maneiras de gerenciar senhas de usuário sem interação humana.

O problema

Queremos construir uma composição de boletim on-line e sistema de envio. Esse sistema deve permitir tanto que os diversos boletins sejam criados e enviados para os usuários como permitir que os usuários assinem, isto é, se inscrevam para receber um ou muitos dos boletins.

Especificamente, os requisitos para esse sistema são:

- Os administradores devem ser capazes de configurar e modificar listas de mala-direta.
- Os administradores devem ser capazes de enviar texto e boletins de HTML para todos os assinantes de uma única lista de mala-direta.
- Os usuários devem ser capazes de registrarem-se para utilizar o site e inserir e modificar seus detalhes.

- Os usuários devem ser capazes de assinar qualquer lista em um site.
- Os usuários devem ser capazes de cancelar a assinatura de listas que assinaram.
- Os usuários devem ser capazes de armazenar suas preferências em HTML formatada ou em boletins de texto simples.
- Por razões de segurança, usuários não devem ser capazes de enviar correio às listas ou de ver endereços de e-mail
- Os usuários e administradores devem ser capazes de visualizar informações sobre lista de mala-direta.
- Os usuários e administradores devem ser capazes de visualizar boletins passados que foram enviados para uma lista (o repositório).

Componentes da solução

Há vários componentes necessários para atendermos aos requisitos dessa aplicação. Os principais são configurar um banco de dados de listas, assinantes e boletins em repositórios; fazer upload de boletins que foram criados off-line; e enviar correio com anexos.

Configurando um banco de dados de listas e assinantes

Monitoraremos o nome de usuário e a senha de cada usuário do sistema, bem como uma lista das listas que eles assinaram. Além disso armazenaremos a preferência de cada usuário em receber e-mail como texto simples ou HTML, de modo que possamos enviar a versão apropriada do boletim para um usuário.

Um administrador será um usuário especializado com a capacidade de criar novas listas de mala-direta e enviar boletins para essas listas.

Uma funcionalidade interessante de se ter em um sistema como esse é um repositório de boletins anteriores. Os assinantes talvez não mantenham as mensagens anteriores, mas podem querer procurar algo. Um repositório também pode atuar como uma ferramenta de marketing para o boletim uma vez que assinantes potenciais podem ver como são os boletins.

Configurar esse banco de dados no MySQL e uma interface para ele no PHP não será nada novo nem difícil.

Usando upload de arquivo

Precisamos de uma interface para permitir ao administrador enviar boletins, como mencionado anteriormente. O que não discutimos é a maneira como os administradores criarão esse boletim. Poderíamos fornecer a eles um formulário no qual poderiam digitar ou colar o conteúdo do boletim. Mas, isso aumentará a característica amigável ao usuário do nosso sistema para permitir que os administradores criem um boletim no editor favorito e então façam upload do arquivo para o servidor Web. Isso também facilitará a um administrador adicionar imagens a um boletim de HTML. Para isso podemos utilizar a capacidade de upload de arquivo discutida no Capítulo 18.

Precisaremos utilizar um formulário ligeiramente mais complexo do que o utilizado antes. Exigiremos que o administrador faça upload tanto da versão texto como da versão HTML do boletim, junto com qualquer imagem embutida que entre na HTML.

Depois que o boletim for carregado, precisamos criar uma interface para que o administrador possa visualizá-lo antes de enviar. Dessa maneira, ele pode confirmar que todos os arquivos foram carregados corretamente.

Observe que armazenaremos todos esses arquivos em um diretório repositório de arquivos, de modo que os usuários possam ler matérias antigas dos boletins. Esse diretório precisa ser gravável

pelo usuário sob o qual seu servidor Web é executado. O script de carregamento tentará gravar os boletins em ./archive/; portanto, certifique-se de criar e configurar permissões para esse diretório apropriadamente.

Enviando correio com anexos

Para esse projeto, queremos ser capazes de enviar aos usuários um boletim simples de texto ou uma versão "sofisticada" em HTML, de acordo com a preferência deles.

Para enviar um arquivo HTML com imagens embutidas, precisaremos descobrir uma maneira de enviar anexos. A simples função `mail( )` do PHP não suporta facilmente enviar anexos. Em vez disso, utilizaremos o excelente pacote `Mail_Mime` do PEAR, criado por Richard Heyes. Esse pacote pode administrar anexos de HTML e também pode ser utilizado para anexar quaisquer imagens contidas no arquivo HTML.

As instruções de instalação desse pacote estão incluídas no Apêndice A.

Visão geral da solução

Para este projeto, novamente utilizaremos uma abordagem baseada em evento para escrever o código, como fizemos no Capítulo 29.

Começamos novamente elaborando um conjunto de diagramas de fluxo de sistema para mostrar aos usuários caminhos a tomar no sistema. Nesse caso, desenhamos três diagramas para representar os três diferentes conjuntos de interações que os usuários terão com o sistema. Os usuários têm diferentes ações admissíveis quando não estão conectados, quando estão conectados como usuários regulares e quando estão conectados como administradores. Essas ações são mostradas nas Figuras 30.1, 30.2 e 30.3, respectivamente.

Na Figura 30.1, você pode ver as ações que podem ser tomadas por um usuário que não está conectado. Como você pode ver, ele pode efetuar logon (se já tiver uma conta), criar uma conta (se ainda não tiver uma) ou visualizar as listas de mala-direta disponíveis para assinatura (como uma tática de marketing).

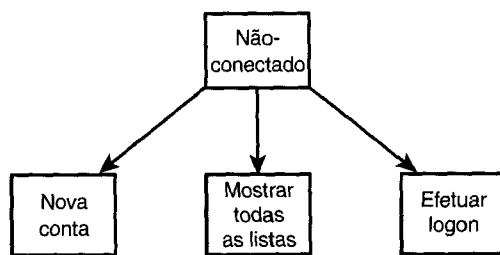



Figura 30.1 Um usuário pode escolher apenas um número limitado de ações quando não estiver conectado.

A Figura 30.2 mostra as ações que um usuário pode tomar depois de efetuar logon. Você pode alterar a configuração de conta do usuário (endereço de e-mail e preferências), alterar sua senha e alterar a lista de que ele é assinante.

A Figura 30.3 mostra as ações disponíveis se um administrador tiver efetuado logon. Como você pode ver, um administrador tem a maior parte da funcionalidade disponível para um usuário e algumas opções adicionais. Ele também pode criar novas listas de mala-direta, criar novas mensagens para uma lista de mala-direta fazendo upload de arquivos e visualizar mensagens antes de enviá-las.

Como utilizamos outra vez uma abordagem baseada em evento, a espinha dorsal da aplicação está contida em um arquivo, `index.php`, que chama um conjunto de bibliotecas de funções. Uma visão geral dos arquivos nessa aplicação é mostrada na Tabela 30.1.

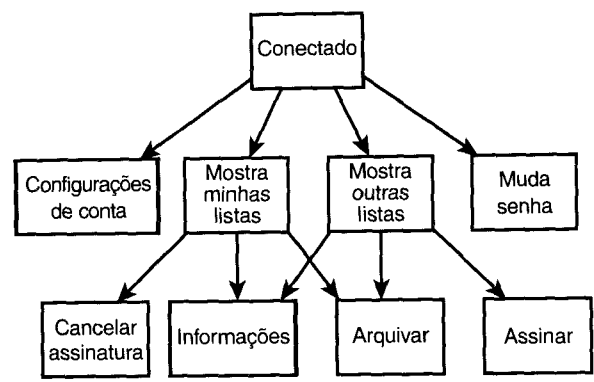



Figura 30.2 Depois de efetuar logon, os usuários podem alterar suas preferências por meio de uma variedade de opções.

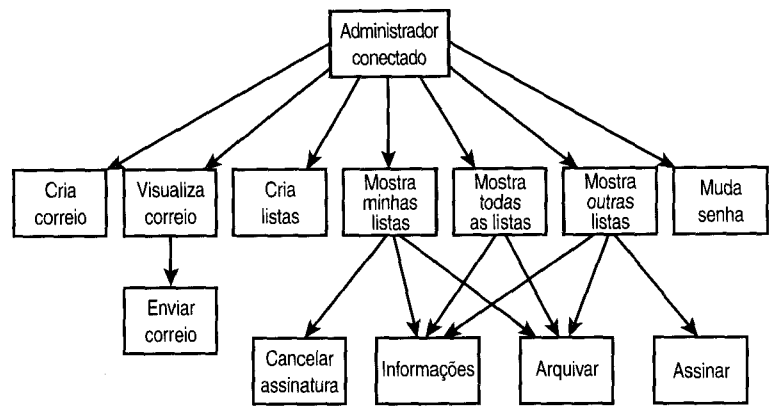



Figura 30.3 Os administradores têm ações adicionais disponíveis para eles.

Tabela 30.1 **Arquivos na aplicação de gerenciamento de listas de mala-direta**

Nome de arquivo	Tipo	Descrição
`index.php`	Aplicação	O script principal que executa a aplicação inteira.
`include_fns.php`	Funções	Coleção de arquivos include para essa aplicação.
`data_valid_fns.php`	Funções	Coleção de funções para validar os dados de entrada.
`db_fns.php`	Funções	Coleção de funções para conectar-se ao banco de dados `mlm`.
`mlm_fns.php`	Funções	Coleção de funções específicas a essa aplicação.
`output_fns.php`	Funções	Coleção de funções para gerar a saída em HTML.
`upload.php`	Componente	Script que gerencia componente de upload de arquivo do papel de administrador. Separado para facilitar a segurança.
`user_auth_fns.php`	Funções	Coleção de funções para autenticar usuários.
`create_database.sql`	SQL	SQL para configurar o banco de dados `mlm` e configurar um usuário Web e um usuário administrativo.

Percorreremos passo a passo o processo de implementação do projeto, começando com o banco de dados em que armazenaremos as informações do assinante e da lista.

Configurando o banco de dados

Para essa aplicação precisaremos armazenar detalhes de:

- **Listas:** Listas de mala-direta disponíveis para assinatura.
- **Assinantes:** Usuários do sistema e suas preferências.
- **Sublistas:** Um registro de quais usuários assinaram quais listas (relacionamento de um muitos para muitos).
- **Correio:** Um registro de mensagens de e-mail que foram enviadas.
- **Imagens:** Como queremos ser capazes de enviar mensagens de e-mail que consistam em diversos arquivos (isto é, texto e HTML mais um número de imagens), também precisamos monitorar que imagens vão com cada e-mail.

A SQL que escrevemos para criar esse banco de dados é mostrada na Listagem 30.1.

Listagem 30.1 **`create_database.sql` – SQL para criar o banco de dados `mlm`**

```
create database mlm;

use mlm;

create table lists
(
   listid int auto_increment not null primary key,
   listname char(20) not null,
   blurb varchar(255)
);

create table subscribers
(
  email char(100) not null primary key,
  realname char(100) not null,
  mimetype char(1) not null,
  password char(16) not null,
  admin tinyint not null
);

# armazena um relacionamento entre um assinante e uma lista
create table sub_lists
(
  email char(100) not null,
  listid int not null
);

create table mail
(
  mailid int auto_increment not null primary key,
  email char(100) not null,
  subject char(100) not null,
  listid int not null,
  status char(10) not null,
  sent datetime,
  modified timestamp
);

#armazena a imagem que vai junto com um dado correio
create table images
(
```

Listagem 30.1 **Continuação**

```
  mailid int  not null,
  path char(100) not null,
  mimetype char(100) not null
);

grant select, insert, update, delete
on mlm.*
to mlm@localhost identified by 'password';

insert into subscribers values
('admin@localhost', 'Administrative User', 'H', password('admin'), 1);
```

Lembre-se de que você pode executar essa instrução de SQL digitando:

```
mysql -u root -p < create_database.sql
```

Você precisará fornecer sua senha de root. (Você poderia, é claro, executar esse script via qualquer usuário do MySQL com os privilégios apropriados; utilizamos root aqui por questão de simplicidade.) Você deve mudar a senha para o usuário `mlm` e o administrador no seu script antes de executá-la.

Alguns campos nesse banco de dados exigem uma pequena explicação adicional, então vamos examiná-lo rapidamente.

A tabela `lists` contém um `listid` e `listname`. Essa tabela também contém um `blurb`, que é uma descrição sobre o que é a lista.

A tabela `subscribers` contém endereços de e-mail (`email`) e nomes (`realname`) dos assinantes. Também armazena seu `password` e um flag (`admin`) para indicar se esse usuário é administrador. Também armazenaremos o tipo de correio que eles preferem receber em `mimetype`. Isso pode ser `H`, de HTML, ou `T`, de texto.

A tabela `sublists` contém endereços de e-mail (`email`) da tabela `subscribers` e `listids` da tabela `lists`.

A tabela `mail` contém informações sobre cada mensagem de e-mail que é enviada pelo sistema. Essa tabela armazena um id único (`mailid`), o endereço a partir do qual o correio é enviado (`email`), a linha de assunto do e-mail (`subject`) e o `listid` da lista para a qual ele foi ou será enviado. O texto real ou HTML da mensagem poderia ser um arquivo grande, então armazenaremos o repositório das mensagens reais fora do banco de dados. Além disso, monitoraremos algumas informações gerais de status: se a mensagem foi enviada (`status`), quando foi enviada (`sent`) e um registro de data/hora para mostrar quando esse registro foi modificado pela última vez (`modified`).

Por fim, utilizamos a tabela `images` para monitorar qualquer imagem associada às mensagens de HTML. Novamente, essas imagens podem ser grandes, então as armazenaremos fora do banco de dados por questão de eficiência. Em vez disso, monitoraremos o `mailid` a que eles são associados, o caminho (`path`) para a localização em que a imagem realmente está armazenada e o tipo da imagem MIME (`mimetype`), por exemplo, `image/gif`.

A SQL mostrada anteriormente também configura a maneira como um usuário se conecta ao PHP e a maneira como um usuário administrativo se conecta ao sistema.

Definindo a arquitetura do script

Como no último projeto, utilizamos uma abordagem baseada em evento para este projeto. A espinha dorsal da aplicação está no arquivo `index.php`. Esse script tem quatro segmentos principais, que são:

1. Pré-processar: Faz qualquer processamento que deve ser feito antes de os cabeçalhos serem enviados.

2. Configurar e enviar cabeçalhos: Cria e envia o início da página HTML.
3. Realizar uma ação: Responde ao evento que foi passado. Como no nosso último exemplo, o evento é contido na variável `$action`.
4. Enviar rodapés.

Quase todo o processamento da aplicação é feito nesse arquivo. A aplicação também utiliza as bibliotecas de funções listadas na Tabela 30.1, como mencionado anteriormente.

A listagem completa do script `index.php` é mostrada na Listagem 30.2.

Listagem 30.2 **`index.php`** – **Arquivo principal de aplicação para Pyramid-MLM**

```
<?php

/**********************************************************************
* Seção 1 : pré-processamento
**********************************************************************/

  require_once ('include_fns.php');
  session_start( );

  $action = $_GET['action'];
  $buttons = array( );

  //acrescenta a essa string se algo for processado antes da saída do cabeçalho
  $status = '';

  // precisa processar pedidos de logon ou logout antes de qualquer coisa
  if($_POST['email']&&$_POST['password'])
  {
    $login = login($_POST['email'], $_POST['password']);

    if($login == 'admin')
    {
      $status .= "<p><b>".get_real_name($_POST['email']).
                 "</b> logged in"." successfully as <b>Administrator</b></p>
                  <br /><br /><br /><br /><br />";
      $_SESSION['admin_user'] = $_POST['email'];
    }
    else if($login == 'normal')
    {
      $status .= "<p><b>".get_real_name($_POST['email'])."</b> logged in"
                 ." successfully.</p><br /><br />";
      $_SESSION['normal_user'] = $_POST['email'];
    }
    else
    {
      $status .= "<p>Sorry, we could not log you in with that
                  email address and password.</p><br />";
    }
  }

  if($action == 'log-out')
  {
    unset($action);
    unset($_SESSION);
    session_destroy( );
  }
```

Listagem 30.2 **Continuação**

```
/*********************************************************************
* Seção 2: configura e exibe cabeçalhos
*********************************************************************/

 // configura os botões que estarão na barra de ferramentas
 if(check_normal_user( ))
 {
   // se um usuário normal
   $buttons[0] = 'change-password';
   $buttons[1] = 'account-settings';
   $buttons[2] = 'show-my-lists';
   $buttons[3] = 'show-other-lists';
   $buttons[4] = 'log-out';
 }
 else if(check_admin_user( ))
 {
   // se um administrador
   $buttons[0] = 'change-password';
   $buttons[1] = 'create-list';
   $buttons[2] = 'create-mail';
   $buttons[3] = 'view-mail';
   $buttons[4] = 'log-out';
   $buttons[5] = 'show-all-lists';
   $buttons[6] = 'show-my-lists';
   $buttons[7] = 'show-other-lists';
 }
 else
 {
   // se não estiver conectado
   $buttons[0] = 'new-account';
   $buttons[1] = 'show-all-lists';
   $buttons[4] = 'log-in';
 }

 if($action)
 {
   // exibe cabeçalho com o nome da aplicação e descrição da página ou ação
   do_html_header('Pyramid-MLM - '.format_action($action));
 }
 else
 {
   // exibe cabeçalho apenas com o nome da aplicação
   do_html_header('Pyramid-MLM');
 }

 display_toolbar($buttons);

 //exibe qualquer texto gerado por funções chamadas antes do cabeçalho
 echo $status;

/*********************************************************************
* Seção 3: realiza a ação
*********************************************************************/

 // só podem ser feitas essas ações se não estiver conectado
 switch ( $action )
 {
   case 'new-account' :
   {
     // livra-se das variáveis de sessão
```

Listagem 30.2 Continuação

```
    session_destroy( );
    display_account_form( );
    break;
  }
  case 'store-account' :
  {
    if (store_account($_SESSION['normal_user'],
            $_SESSION['admin_user'], $_POST))
      $action = '';
    if(!check_logged_in( ))
      display_login_form($action);
    break;
  }
  case 'show-all-lists' :
  {
    display_items('All Lists', get_all_lists( ), 'information',
                  'show-archive','');
    break;
  }
  case 'show-archive':
  {
    display_items('Archive For '.get_list_name($_GET['id']),
                   get_archive($_GET['id']), 'view-html',
                  'view-text', '');
    break;
  }
  case 'information' :
  {
    display_information($_GET['id']);
    break;
  }
  default :
  {
    if(!check_logged_in( ))
      display_login_form($action);
    break;
  }
}

//todas as outras ações requerem que o usuário esteja conectado
if(check_logged_in( ))
{
  switch ( $action )
  {
    case 'account-settings' :
    {
      display_account_form(get_email( ),
            get_real_name(get_email( )), get_mimetype(get_email( )));
      break;
    }
    case 'show-other-lists' :
    {
      display_items('Unsubscribed Lists',
                    get_unsubscribed_lists(get_email( )), 'information',
                    'show-archive', 'subscribe');
      break;
    }
    case 'subscribe' :
    {
      subscribe(get_email( ), $_GET['id']);
```

Listagem 30.2 **Continuação**

```
        display_items('Subscribed Lists', get_subscribed_lists(get_email( )),
                   'information', 'show-archive', 'unsubscribe');
        break;
      }
      case 'unsubscribe' :
      {
        unsubscribe(get_email( ), $_GET['id']);
        display_items('Subscribed Lists', get_subscribed_lists(get_email( )),
                   'information', 'show-archive', 'unsubscribe');
        break;
      }
      case '':
      case 'show-my-lists' :
      {
        display_items('Subscribed Lists', get_subscribed_lists(get_email( )),
                   'information', 'show-archive', 'unsubscribe');
        break;
      }
      case 'change-password' :
      {
        display_password_form( );
        break;
      }
      case 'store-change-password' :
      {
        if(change_password(get_email( ), $_POST['old_passwd'],
           $_POST['new_passwd'], $_POST['new_passwd2']))
        {
          echo '<p>OK: Password changed.</p>
                <br /><br /><br /><br /><br /><br />';
        }
        else
        {
          echo '<p>Sorry, your password could not be changed.</p>';
          display_password_form( );
        }
        break;
      }
    }
  }
  // As seguintes ações podem apenas ser realizadas por um usuário admin
  if(check_admin_user( ))
  {
    switch ( $action )
    {
      case 'create-mail' :
      {
        display_mail_form(get_email( ));
        break;
      }
      case 'create-list' :
      {
        display_list_form(get_email( ));
        break;
      }
      case 'store-list' :
      {
        if(store_list($_SESSION['admin_user'], $_POST))
        {
          echo '<p>New list added</p><br />';
```

Listagem 30.2 **Continuação**

```
        display_items('All Lists', get_all_lists( ), 'information',
                      'show-archive','');
      }
      else
        echo '<p>List could not be stored, please try '
             .'again.</p><br /><br /><br /><br /><br />';

      break;
    }
    case 'send' :
    {
      send($_GET['id'], $_SESSION['admin_user']);
      break;
    }
    case 'view-mail' :
    {
      display_items('Unsent Mail', get_unsent_mail(get_email( )),
                    'preview-html', 'preview-text', 'send');
      break;
    }
  }
}

/**********************************************************************
* Seção 4: exibe rodapé
*********************************************************************/

  do_html_footer( );
? >
```

Você pode ver os quatro segmentos do código claramente marcados nessa listagem.

Na etapa de pré-processamento, configuramos a sessão e processamos qualquer ação que precisa ser feita antes que os cabeçalhos possam ser enviados. Nesse caso, isso inclui efetuar logon e logout.

Na etapa de cabeçalho, configuramos os botões de menu que o usuário verá e exibimos os cabeçalhos apropriados utilizando a função `do_html_header( )` de `output_fns.php`. Essa função só exibe a barra de cabeçalho e os menus, então não a abordaremos aqui.

Na seção principal do script, respondemos à ação que o usuário escolheu. Essas ações são divididas em três subconjuntos: ações que podem ser tomadas por usuários não-conectados, ações que podem ser tomadas por usuários normais e ações que podem ser tomadas por usuários administrativos. Verificamos para ver se o acesso aos dois últimos conjuntos de ações é permitido utilizando as funções `check_logged_in( )` e `check_admin_user( )`. Essas funções encontram-se na biblioteca de funções `user_auth_fns.php`. O código para as funções e para a função `check_normal_user( )` é mostrado na Listagem 30.3.

Listagem 30.3 **Funções de `user_auth_fns.php` – Verificam se um usuário está ou não conectado e em que nível**

```
function check_normal_user( )
// verifica se alguém está conectado e notifica se não estiver
{
 if (isset($_SESSION['normal_user']))
   return true;
 else
   return false;
}
```

Listagem 30.3 **Continuação**

```
function check_admin_user( )
// verifica se alguém está conectado e notifica se não estiver
{
 if (isset($_SESSION['admin_user']))
   return true;
 else
   return false;
}

function check_logged_in( )
{
 return ( check_normal_user( ) || check_admin_user( ) );
}
```

Como você pode ver, essas funções utilizam as variáveis de sessão `$normal_user` e `$admin_user` para verificar se um usuário se conectou. Discutiremos como configurar essas variáveis de sessão em um minuto.

Na seção final do script, enviamos um rodapé de HTML utilizando a função `do_html_footer( )` de `output_fns.php`.

Vamos examinar resumidamente as possíveis ações no sistema. Essas ações são mostradas na Tabela 30.2

Tabela 30.2 **Possíveis ações na aplicação de gerenciamento de listas de mala-direta**

Ação	Usável Por	Descrição
`log-in`	Qualquer pessoa	Fornece um formulário de login para um usuário.
`log-out`	Qualquer pessoa	Termina uma sessão.
`new-account`	Qualquer pessoa	Cria uma nova conta para um usuário.
`store-account`	Qualquer pessoa	Armazena detalhes da conta.
`show-all-lists`	Qualquer pessoa	Mostra uma lista de listas de mala-direta disponíveis.
`show-archive`	Qualquer pessoa	Exibe boletins com repositórios para uma lista particular.
`information`	Qualquer pessoa	Mostra informações básicas sobre uma lista particular.
`account-settings`	Usuários conectados	Exibe configurações da conta do usuário.
`show-other-lists`	Usuários conectados	Exibe as listas de mala-direta de que o usuário não é assinante.
`show-my-lists`	Usuários conectados	Exibe as listas de mala-direta de que o usuário é assinante.
`subscribe`	Usuários conectados	Inscreve um usuário em uma lista particular.
`unsubscribe`	Usuários conectados	Cancela a assinatura de um usuário em uma lista particular.
`change-password`	Usuários conectados	Exibe o formulário de alteração de senha.
`store-change-password`	Usuários conectados	Atualiza a senha do usuário no banco de dados.
`create-mail`	Administradores	Exibe formulário para permitir upload de boletins.
`create-list`	Administradores	Exibe formulário para permitir que novas listas de mala-direta sejam criadas.

Tabela 30.2 **Continuação**

Ação	Usável Por	Descrição
store-list	Administradores	Armazena detalhes de lista de mala-direta no banco de dados.
view-mail	Administradores	Exibe boletins que foram carregados mas ainda não foram enviados.
Send	Administradores	Envia boletins para assinantes.

Uma omissão notável dessa tabela é uma opção ao longo das linhas de `store-mail`, isto é, uma ação que realmente faça upload de boletins inseridos via `create-mail` por administradores. Essa única parte de funcionalidade está, na realidade, em um arquivo diferente, `upload.php`. Colocamos essa parte em um arquivo separado porque isso facilita um pouco para nós, programadores, monitorarmos questões de segurança.

A seguir, discutiremos a implementação dessas ações nos três grupos listados na Tabela 30.2, isto é, ações para pessoas que não estão conectadas, ações para usuários conectados e ações para administradores.

Implementando login

Quando um usuário visita nosso site pela primeira vez, há três ações que gostaríamos que ele fizesse. Primeiro, ver o que temos a oferecer; segundo, fazer uma assinatura conosco; e terceiro, efetuar logon.

Na Figura 30.4, você pode ver a tela que apresentamos aos usuários quando eles chegam ao nosso site pela primeira vez.

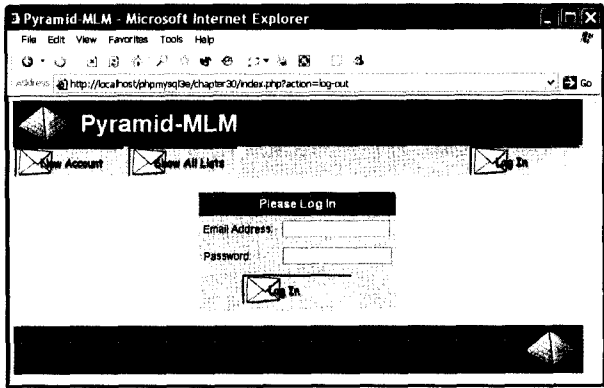



Figura 30.4 Na chegada, os usuários podem criar uma nova conta, visualizar listas disponíveis ou simplesmente efetuar logon.

Agora, veremos como criar uma nova conta e efetuar logon e retornaremos para ver os detalhes da listagem nas seções "Implementando funções de usuário" e "Implementando funções administrativas" mais adiante.

Criando uma nova conta

Se um usuário selecionar a opção de menu New Account, isso ativa a ação `new-account`. Isso ativa o seguinte código no `index.php`:

```
case 'new-account' :
  {
    // livra-se das variáveis de sessão
```

```
    session_destroy( );
    display_account_form( );
    break;
  }
```

Esse código efetivamente realiza o logout de um usuário se ele estiver atualmente conectado e exibe o formulário de detalhes da conta como é mostrado na Figura 30.5.

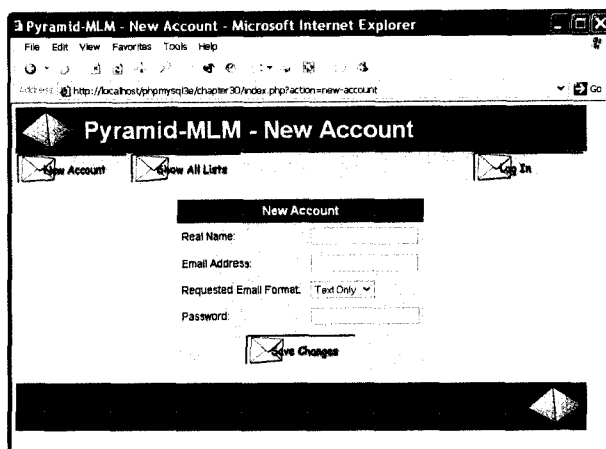



Figura 30.5 O novo formulário de criação de conta permite que os usuários insiram seus detalhes.

Esse formulário é gerado pela função `display_account_form( )` da biblioteca `output_fns.php`. Essa função é utilizada tanto aqui como na ação de configuração de contas para exibir um formulário e permitir que o usuário configure uma conta. Se a função for invocada a partir da ação de configuração de contas, o formulário será preenchido com os dados da conta existente do usuário. Aqui o formulário está em branco, pronto para novos detalhes da conta. Como essa função envia somente saída para HTML, não a examinaremos em detalhe aqui.

O botão submit nesse formulário invoca a ação `store-account`. O código para essa ação é o seguinte:

```
case 'store-account' :
{
 if (store_account($_SESSION['normal_user'],
        $_SESSION['admin_user'], $_POST))
   $action = '';
 if(!check_logged_in( ))
   display_login_form($action);
 break;
}
```

A função `store_account( )` grava os detalhes da conta no banco de dados. O código para essa função é mostrado na Listagem 30.4.

Listagem 30.4 **Função `store_account( )` de `mlm_fns.php` – Essa função adiciona um novo usuário ou armazena detalhes modificados sobre um usuário existente**

```
function store_account($normal_user, $admin_user, $details)
{
  if(!filled_out($details))
  {
    echo 'All fields must be filled in.  Try again.<br /><br />';
    return false;
```

Listagem 30.4 **Continuação**

```
}
 else
 {
   if(subscriber_exists($details['email']))
   {
     //verifica login como o usuário que estão tentando mudar
     if(get_email( )==$details['email'])
     {
       $query = "update subscribers set realname = '{$details['realname']}',
                                        mimetype = '{$details['mimetype']}'
                 where email = '{$details['email']}'";
       if($conn=db_connect( ))
       {
         if ($conn->query($query))
           return true;
         else
           return false;
       }
       else
       {
         echo 'Could not store changes.<br /><br /><br /><br /><br /><br />';
         return false;
       }
     }
     else
     {
       echo '<p>Sorry, that email address is already registered here.</p>'.
            '<p>You will need to log in with that address to change '.
            ' its settings.</p>';
       return false;
     }
   }
   else // new account
   {
     $query = "insert into subscribers
               values ('{$details['email']}',
                       '{$details['realname']}',
                       '{$details['mimetype']}',
                       sha1('{$details['new_password']}'),
                       0)";
     if($conn=db_connect( ))
     {
      if ($conn->query($query))
        return true;
      else
        return false;
     }
     else
     {
       echo 'Could not store new account.
             <br /><br /><br /><br /><br /><br />';
       return false;
     }
   }
 }
}
```

Essa função primeiro verifica se o usuário preencheu os detalhes exigidos.

Se isso estiver ok, a função então criará um novo usuário ou atualizará os detalhes da conta se o usuário já existir. Um usuário só pode atualizar os detalhes da conta do usuário com que ele está conectado.

Isso é verificado utilizando a função get_email( ), que recupera o endereço de e-mail do usuário que está atualmente conectado. Retornaremos para essa função mais tarde, já que ela utiliza as variáveis de sessão que são configuradas quando o usuário efetua logon.

Efetuando logon

Se um usuário preencher o formulário de login que vimos na Figura 30.4 e clicar no botão Log In, ele entrará no script index.php com as variáveis email e password configuradas. Isso ativará o código de login, que está na etapa de pré-processamento do script, da seguinte maneira:

```
// precisa processar pedidos de logon ou logout antes de qualquer coisa
if($_POST['email']&&$_POST['password'])
{
  $login = login($_POST['email'], $_POST['password']);

  if($login == 'admin')
  {
    $status .= "<p><b>".get_real_name($_POST['email']).
               "</b> logged in"." successfully as <b>Administrator</b></p>
               <br /><br /><br /><br /><br />";
    $_SESSION['admin_user'] = $_POST['email'];
  }
  else if($login == 'normal')
  {
    $status .= "<p><b>".get_real_name($_POST['email'])."</b> logged in"
               ." successfully.</p><br /><br />";
    $_SESSION['normal_user'] = $_POST['email'];
  }
  else
  {
    $status .= "<p>Sorry, we could not log you in with that
                email address and password.</p><br />";
  }
}

if($action == 'log-out')
{
  unset($action);
  unset($_SESSION);
  session_destroy( );
}
```

Como você pode ver, primeiro tentamos fazer com que eles efetuem o login utilizando a função login( ) da biblioteca user_auth_fns.php. Isso é um pouco diferente das funções de login que utilizamos em outra parte; portanto, iremos examiná-la. O código para essa função é mostrado na Listagem 30.5.

Listagem 30.5 **Função login( ) de user_auth_fns.php –Verificando os detalhes de login de um usuário**

```
function login($email, $password)
// verifica nome de usuário e senha no bd
// se sim, retorna tipo de login
// caso contrário retorna false
{
```

Listagem 30.5 **Continuação**

```
  // conecta-se ao bd
  $conn = db_connect( );
  if (!$conn)
    return 0;

  $query = "select admin from subscribers
                          where email='$email'
                         and password = password('$password')";
  //eco $query;
  $result = mysql_query($query);
  if (!$result)
    return false;

  if (mysql_num_rows($result)<1)
    return false;

  if(mysql_result($result, 0, 0) == 1)
    return 'admin';
  else
    return 'normal';
}
```

Anteriormente com as funções de login, retornávamos `true` se o login era bem-sucedido e `false`, caso contrário. Nesse caso, ainda retornamos `false` se o login falhar, mas se fosse bem-sucedido retornaríamos o tipo de usuário, `'admin'` ou `'normal'`. Verificamos o tipo de usuário recuperando o valor armazenado na coluna `admin` na tabela dos assinantes, para uma combinação particular de endereço de e-mail e senha. Se nenhum resultado foi retornado, retornamos `false`. Se um usuário for administrador, esse valor será 1 (`true`) e retornamos `'admin'`. Caso contrário, retornamos `'normal'`.

Retornando à linha de execução principal, registramos uma variável de sessão para monitorar quem é nosso usuário. Esse será `admin_user` se for um administrador, ou `normal_user` se for um usuário regular. Quaisquer dessas variáveis que configuramos conterá o endereço de e-mail do usuário. Para simplificar a verificação do endereço de e-mail de um usuário, utilizamos a função `get_email( )` mencionada anteriormente.

Essa função é mostrada na Listagem 30.6.

Listagem 30.6 **get_email( ) de user_auth_fns.php – Retorna o endereço de e-mail do usuário conectado**

```
function get_email( )
{
  if (isset($_SESSION['normal_user']))
    return $_SESSION['normal_user'];
  if (isset($_SESSION['admin_user']))
   return $_SESSION['admin_user'];

  return false;
}
```

De volta ao nosso programa principal, informamos ao usuário se ele foi conectado ou não e em que nível.

A saída de uma tentativa de login é mostrada na Figura 30.6.

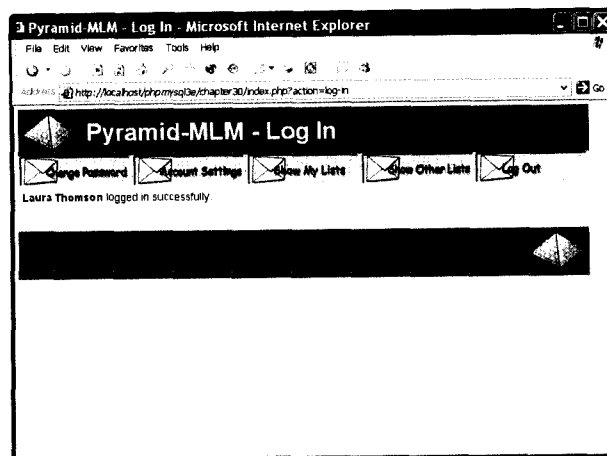



Figura 30.6 O sistema informa ao usuário que o login foi bem-sucedido.

Agora que conectamos um usuário, podemos prosseguir às funções de usuário.

Implementando funções de usuário

Há cinco ações que desejamos que nossos usuários possam fazer depois que efetuarem logon:

- Ver as listas disponíveis para assinatura;
- Fazer e cancelar assinatura de listas;
- Alterar a maneira como suas contas são configuradas;
- Alterar suas senhas;
- Efetuar logout.

Você pode ver a maioria dessas opções na Figura 30.6. Agora veremos a implementação de cada uma dessas opções.

Visualizando listas

Neste projeto, implementaremos várias opções para visualizar listas disponíveis e detalhes de lista. Na Figura 30.6, você pode ver duas dessas opções: Show My Lists, que recuperará as listas que esse usuário assinou e Show Other Lists, que recuperará as listas que o usuário não assinou.

Se examinar outra vez a Figura 30.4, verá que temos outra opção, Show All Lists, que recuperará todas as listas de mala-direta disponíveis no sistema. Para o sistema ser verdadeiramente escalonável, devemos adicionar funcionalidades de paginação (exibir, digamos, 10 resultados por página). Não fizemos isso aqui para sermos mais breves.

Essas três opções de menu ativam as ações `show-my-lists`, `show-other-lists` e `show-all-lists`, respectivamente. Como você provavelmente percebeu, todas essas ações funcionam de maneira semelhante. O código para essas três ações é:

```
case 'show-all-lists' :
{
   display_items('All Lists', get_all_lists( ), 'information',
                 'show-archive','');
   break;
}


case 'show-other-lists' :
{
```

```
      display_items('Unsubscribed Lists',
                    get_unsubscribed_lists(get_email( )), 'information',
                    'show-archive', 'subscribe');
      break;
   }

   case '':
   case 'show-my-lists' :
   {
      display_items('Subscribed Lists', get_subscribed_lists(get_email( )),
                   'information', 'show-archive', 'unsubscribe');
      break;
   }
```

Como você pode ver, todas essas ações chamam a função display_items( ) da biblioteca output_fns.php, mas cada uma delas chama essa função com parâmetros diferentes. Todas elas utilizam também a função get_email( ), que mencionamos anteriormente, para obter o endereço apropriado de e-mail para esse usuário.

Para ver o que essa função faz, veja a Figura 30.7, que mostra a página Show Other Lists.

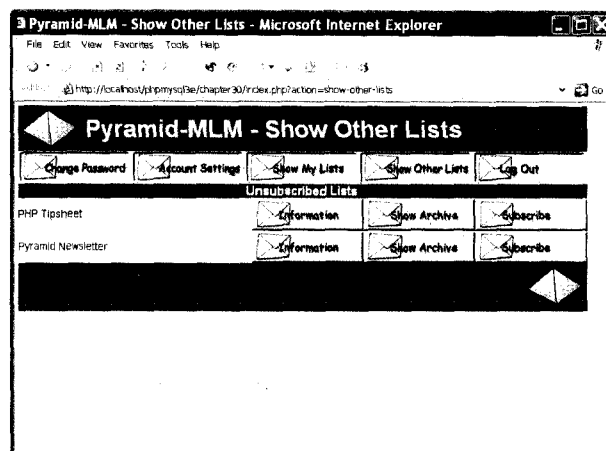



Figura 30.7 A função display_items( ) foi utilizada para compor uma lista das listas de que o usuário não é assinante.

Vamos ver o código para a função display_items( ). Esse código é mostrado na Listagem 30.7.

Listagem 30.7 **Função display_items( ) de output_fns.php – Essa função é utilizada para exibir uma lista de itens com ações associadas**

```
function display_items($title, $list, $action1='', $action2='', $action3='')
{
   global $table_width;
   echo "<table width=\"$table_width\" cellspacing=\"0\" cellpadding=\"0\"
          border=\"0\">";

   // número da contagem de ações
   $actions = (($action1!='') + ($action2!='') + ($action3!=''));

   echo '<tr>
         <th colspan="'. (1+$actions) .'" bgcolor="#5B69A6">'. $title .'</th>
         </tr>';

   // número da contagem de itens
   $items = sizeof($list);
```

Listagem 30.7 **Continuação**

```
  if($items == 0)
     echo '<tr>
          <td colspan="'.(1+$actions).'" align="center">No Items to Display</td>
          </tr>';
  else
  {
     // imprime cada linha
     for($i = 0; $items; $i++)
     {
       if($i%2) // cores de fundo alternadas
         $bgcolor = "'#ffffff'";
       else
         $bgcolor = "'#ccccff'";
       echo "<tr>
             <td bgcolor=$bgcolor
              width=\"".($table_width - ($actions*149)).'">';
       echo $list[$i][1];
       if($list[$i][2])
         echo ' - '.$list[$i][2];
       echo '</td>';

       // cria botões para até três ações por linha
       for($j = 1; $j<=3; $j++)
       {
         $var = 'action'.$j;
         if($$var)
         {
           echo "<td bgcolor=\"$bgcolor\" width=\"149\">";
           // botões view/preview são um caso especial já que fornecem um link para um arquivo
           if($$var == 'preview-html'||$$var == 'view-html'||
              $$var == 'preview-text'||$$var == 'view-text')
             display_preview_button($list[$i][3], $list[$i][0], $$var);
           else
             display_button( $$var, '&id=' . $list[$i][0] );
           echo '</td>';
         }
       }
       echo "</tr>\n";
     }
     echo '</table>';
  }
}
```

Essa função enviará para saída uma tabela de itens, com cada item tendo até três botões de ação associados. A função espera cinco parâmetros, que são, na ordem:

- `$title` é o título que aparece na parte superior da tabela. No caso mostrado na Figura 30.7, estamos passando o título Unsubscribed Lists, como mostrado no trecho de código anteriormente discutido para a ação "Show Other Lists".

- `$list` é um array de itens para exibir em cada linha da tabela. Nesse caso, é um array das listas de que usuário não é assinante. Estamos construindo esse array (neste caso) na função `get_unsubscribed_lists( )`, que discutiremos em um minuto. Esse é um array multidimensional, com cada linha no array contendo até quatro partes de dados sobre cada linha. Em ordem, novamente:

  - `$list[`*n*`][0]` deve conter o id de item, que normalmente será um número de linha. Isso fornece o id da linha para os botões de ação sobre os quais eles estão operando. Em nosso caso, utilizaremos ids do banco de dados – informações adicionais sobre isso em um minuto.

- $list[*n*][1] deve conter o nome de item. Isso será o texto exibido para um item particular. Por exemplo, no caso mostrado na Figura 30.7, o nome de item na primeira linha da tabela é PHP Tipsheet.
- $list[n][2] e $list[n][3] são opcionais. Utilizamos-os para comunicar que há informações adicionais. Esses parâmetros correspondem ao texto e ao id de mais informações, respectivamente. Veremos um exemplo que utiliza esses dois parâmetros quando examinarmos a ação View Mail na seção "Implementando funções administrativas".

- O terceiro, quarto e quinto parâmetros opcionais para a função são utilizados para passar três ações que serão exibidas nos botões correspondentes para cada item. Na Figura 30.7, esses são os três botões de ação mostrados como Information, Show Archive e Subscribe.

Obtivemos esses três botões para a página Show All Lists passando os nomes de ação, information, show-archive e subscribe. Utilizando a função display_button( ), essas ações transformam-se em botões com essas palavras sobre eles e a ação apropriada atribuída a eles.

Todas as ações Show chamam a função display_items( ) de maneira diferente, como você pode ver examinando novamente suas ações. Além de ter títulos e botões de ação diferentes, cada um dos três utiliza uma função diferente para construir o array de itens a exibir. Show All Lists utiliza a função get_all_lists( ); Show Other Lists utiliza a função get_unsubscribed_lists( ), e Show My Lists utiliza a função get_subscribed_lists( ). Todas essas funções funcionam de modo semelhante. Todas elas são da biblioteca de funções mlm_fns.php.

Veremos get_unsubscribed_lists( ) porque ela é o exemplo que seguimos até agora. O código para a função get_unsubscribed_lists( ) é mostrado na Listagem 30.8.

Listagem 30.8 **Função get_unsubscribed_lists( ) de mlm_fns.php – construindo um array de lista de mala-direta de que um usuário não é assinante**

```
//obtém as listas em que esse usuário *não* está inscrito
function get_unsubscribed_lists($email)
{
  $list = array( );

  $query = "select lists.listid, listname, email from lists left join sub_lists
                   on lists.listid = sub_lists.listid
                   and email='$email' where email is NULL order by listname";
  if(db_connect( ))
  {
    $result = mysql_query($query);
    if(!$result)
      echo '<p>Unable to get list from database.</p>';
    $num = mysql_numrows($result);
    for($i = 0; $i<$num; $i++)
    {
      array_push($list, array(mysql_result($result, $i, 0),
                  mysql_result($result, $i, 1)));
    }
  }
  return $list;
}
```

Como você pode ver, essa função exige que um endereço de e-mail seja passado para ela. Esse deve ser o endereço de e-mail do assinante com que estamos trabalhando. A função get_subscribed_lists( ) também exige um endereço de e-mail como um parâmetro, mas a função get_all_lists( ) não o exige por razões óbvias.

Considerando o endereço de e-mail de um assinante, nos conectamos ao banco de dados e buscamos todas as listas em que o assinante não se inscreveu. Como de costume para esse tipo de consulta no MySQL, utilizamos uma `LEFT JOIN` para localizar itens sem correspondência, e fazemos um loop pelos resultados e construímos o array linha por linha utilizando a função predefinida `array_push( )`.

Agora que sabemos como essa lista é produzida, vejamos os botões de ação associados a essas telas.

Visualizando informações de lista

O botão Information, mostrado na Figura 30.7, aciona a ação `information`, que é assim:

```
case 'information' :
{
  display_information($_GET['id']);
  break;
}
```

Para ver o que a função `display_information( )` faz, veja a Figura 30.8.

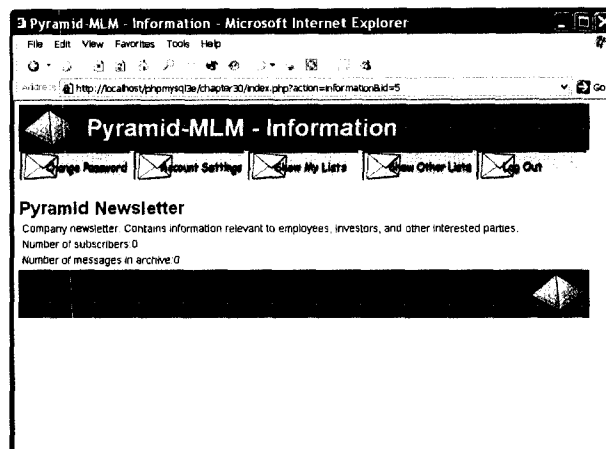



Figura 30.8 A função `display_information( )` mostra uma publicidade de uma lista de mala-direta.

A função exibe algumas informações gerais sobre uma lista de mala-direta particular, além de listar o número de assinantes e o número de boletins que foram enviados para essa lista e estão disponíveis no repositório (informações adicionais em um minuto).

O código para essa função é mostrado na Listagem 30.9.

Listagem 30.9 **Função `display_information( )` de `output_fns.php` – Exibindo informações sobre listas**

```
function display_information($listid)
{
   if(!$listid)
     return false;

   $info = load_list_info($listid);

   if($info)
  {
     echo '<h2>'.pretty($info[listname]).'</h2>';
     echo '<p>'.pretty($info[blurb])';
     echo '<p>Number of subscribers:' . $info[subscribers]'</p>;
     echo '<p>Number of messages in archive:' . $info[archive]'</p>;
   }
}
```

A função display_information( ) utiliza duas outras funções para ajudar a alcançar sua tarefa Web: load_list_info( ) e pretty( ). A função load_list_info( ) realmente recupera os dados do banco de dados. A função pretty( ) simplesmente formata os dados do banco de dados eliminando as barras, transformando caracteres de nova linha em quebras de linha de HTML etc.

Vamos examinar rapidamente a função load_list_info( ). Essa função está na biblioteca de funções mlm_fns.php. O código para essa função é mostrado na Listagem 30.10.

Listagem 30.10 **Função load_list_info( ) de mlm_fns.php – Construindo um array de informações sobre listas**

```
function load_list_info($listid)
{
  if(!$listid)
    return false;

  if(!($conn=db_connect( )))
    return false;

  $query = "select listname, blurb from lists where listid = $listid";
  $result = $conn->query($query);
  if(!$result)
  {
    echo 'Cannot retrieve this list';
    return false;
  }
  $info =  $result->fetch_assoc( );

  $query = "select count(*) from sub_lists where listid = $listid";
  $result = $conn->query($query);
  if($result)
  {
    $row = $result->fetch_array( );
    $info['subscribers'] = $row[0];
  }
  $query = "select count(*) from mail where listid = $listid
            and status = 'SENT'";
  $result = $conn->query($query);
  if($result)
  {
    $row = $result->fetch_array( );
    $info['archive'] = $row[0];
  }
  return $info;
}
```

Essa função executa três consultas ao banco de dados para coletar o nome e a descrição para uma lista a partir da tabela lists; o número de assinantes a partir da tabela sub_lists; e o número de boletins enviados a partir da tabela mail.

Visualizando repositórios de lista

Além de visualizar a descrição da lista, os usuários podem ver todo o correio que foi enviado para uma lista de mala-direta clicando no botão Show Archive. Isso ativa a ação show-archive, que desencadeia o seguinte código:

```
case 'show-archive' :
{
  display_items('Archive For '.get_list_name($_GET['id']),
```

```
                get_archive($_GET['id']), 'view-html',
               'view-text', '');
  break;
}
```

Novamente, essa função utiliza a função `display_items( )` para listar os vários itens de correio que foram enviados para a lista. Esses itens são recuperados utilizando a função `get_archive( )` da biblioteca `mlm_fns.php`. Essa função é mostrada na Listagem 30.11.

Listagem 30.11 **Função `get_archive( )` de `mlm_fns.php` – Construindo um array de boletins arquivados para uma determinada lista**

```
function get_archive($listid)
{
  //retorna um array do correio arquivado para essa lista
  //o array possui linhas como (mailid, subject)

  $list = array( );
  $listname = get_list_name($listid);

  $query = "select mailid, subject, listid
          from mail
          where listid = $listid and status = 'SENT' order by sent";

  if($conn=db_connect( ))
  {
    $result = $conn->query($query);
    if(!$result)
    {
      echo '<p>Unable to get list from database.</p>';
      return false;
    }
    $num = $result->num_rows;

    for($i = 0; $i<$num; $i++)
    {
      $row = $result->fetch_array( );
      $arr_row = array($row[0], $row[1],
                 $listname, $listid);
      array_push($list, $arr_row);
    }
  }
  return $list;
}
```

Novamente, essa função obtém as informações exigidas – nesse caso, os detalhes de correio que foram enviados – a partir do banco de dados e constrói um array adequado para passar à função `display_items( )`.

Assinando e cancelando a assinatura

Na lista de listas de mala-direta mostrada na Figura 30.7, cada lista tem um botão que permite que os usuários façam uma assinatura dela. De maneira semelhante, se os usuários utilizam a opção Show My Lists para ver as listas em que fizeram a assinatura, eles verão um botão Unsubscribe ao lado de cada lista.

Esses botões ativam as ações `subscribe` e `unsubscribe`, que desencadeiam os dois seguintes trechos de código, respectivamente:

```
case 'subscribe' :
{
  subscribe(get_email( ), $_GET['id']);
  display_items('Subscribed Lists', get_subscribed_lists(get_email( )),
                'information', 'show-archive', 'unsubscribe');
  break;
}
case 'unsubscribe' :
{
  unsubscribe(get_email( ), $_GET['id']);
  display_items('Subscribed Lists', get_subscribed_lists(get_email( )),
               'information', 'show-archive', 'unsubscribe');
  break;
}
```

Em cada caso, você chama uma função (subscribe( ) ou unsubscribe( )) e então exibe novamente uma lista das listas de mala-direta de que o usuário é agora assinante utilizando a função display_items( ) novamente.

As funções subscribe( ) e unsubscribe( ) são mostradas na Listagem 30.12.

Listagem 30.12 **Funções subscribe( ) e unsubscribe( ) de mlm_fns.php – Essas funções adicionam e removem assinaturas para um usuário**

```
function subscribe($email, $listid)
{
  if(!$email||!$listid||!list_exists($listid)||!subscriber_exists($email))
    return false;

  //se já assinante, encerra
  if(subscribed($email, $listid))
    return false;

  if(!($conn=db_connect( )))
    return false;

  $query = "insert into sub_lists values ('$email', $listid)";

  $result = $conn->query($query);
  return $result;
}

function unsubscribe($email, $listid)
{
 if(!$email||!$listid)
   return false;

 if(!($conn=db_connect( )))
   return false;

 $query = "delete from sub_lists where email = '$email' and listid = $listid";

 $result = $conn->query($query);
 return $result;
}
```

A função subscribe( ) adiciona uma linha à tabela sub_lists correspondente à assinatura; a função unsubscribe( ) exclui essa linha.

Alterando configurações de conta

O botão Account Settings, quando clicado, ativa a ação `account-settings`. O código para essa ação é:

```
case 'account-settings' :
{
   display_account_form(get_email( ),
         get_real_name(get_email( )), get_mimetype(get_email( )));
   break;
}
```

Como você pode ver, estamos reutilizando a função `display_account_form( )` que utilizamos para criar a conta em primeiro lugar. Entretanto, dessa vez estamos passando os detalhes atuais do usuário, que serão exibidos no formulário para facilitar a edição. Quando o usuário clicar no botão submit desse formulário, a ação `store-account` é ativada como discutido anteriormente.

Alterando senha

Clicar no botão Change Password ativa a ação `change-password`, que desencadeia o seguinte código:

```
case 'change-password' :
{
   display_password_form( );
   break;
}
```

A função `display_password_form( )` (da biblioteca `output_fns.php`) simplesmente exibe um formulário para o usuário alterar sua senha. Esse formulário é mostrado na Figura 30.9.

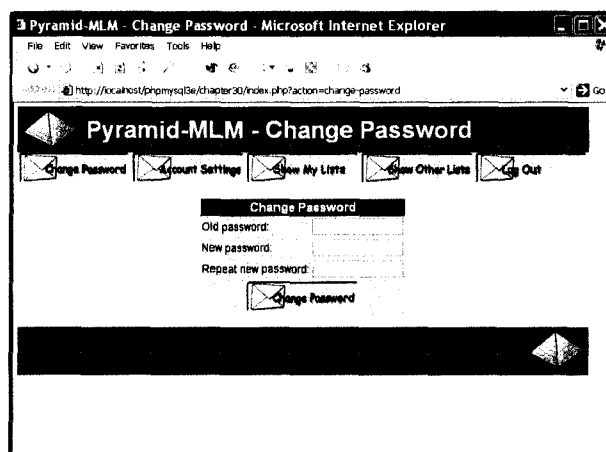



Figura 30.9 A função `display_password_form( )` permite que os usuários alterem suas senhas.

Quando um usuário clica no botão Change Password na parte inferior desse formulário, a ação `store-change-password` é ativada. O código para essa função é:

```
case 'store-change-password' :
{
  if(change_password(get_email( ), $_POST['old_passwd'],
     $_POST['new_passwd'], $_POST['new_passwd2']))
  {
    echo '<p>OK: Password changed.</p>
          <br /><br /><br /><br /><br /><br />';
  }
```

```
  else
  {
    echo '<p>Sorry, your password could not be changed.</p>';
    display_password_form( );
  }
  break;
}
```

Como você pode ver, esse código tenta alterar a senha utilizando a função `change_password( )` e informa ao usuário o sucesso ou falha. A função `change_password( )` pode ser encontrada na biblioteca de funções `user_auth_fns.php`. O código para essa função é mostrado na Listagem 30.13.

Listagem 30.13 **Função `change_password( )` de `user_auth_fns.php` – Essa função valida e atualiza a senha de um usuário**

```
function change_password($email, $old_password, $new_password,
                         $new_password_conf)
// altera senha do e-mail / de old_password para new_password
// retorna true ou false
{

  // se a senha antiga for correta
  // altera sua senha para new_password e retorna true
  // de outra maneira retorna false
  if (login($email, $old_password))
  {
    if($new_password==$new_password_conf)
    {
      if (!($conn = db_connect( )))
        return false;
      $query = "update subscribers
               set password = password('$new_password')
               where email = '$email'";
      $result = mysql_query($query);
      return $result;
    }
    else
      echo '<p> Your passwords do not match.</p>;
  }
  else
    echo '<p> Your old password is incorrect.</p>';

  return false; // a senha antiga estava errada
}
```

Essa função é semelhante às funções de configuração e alteração de senha que examinamos. Essa função compara as duas novas senhas inseridas pelo usuário para certificar-se de que elas são as mesmas, e se forem, tenta atualizar a senha do usuário no banco de dados.

Efetuando Log out

Quando um usuário clica no botão Log Out, ele desencadeia a ação log-out. O código executado por essa ação no script principal está realmente na seção de pré-processamento do script, assim:

```
if($action == 'log-out')
{
  unset($action);
  unset($_SESSION);
  session_destroy( );
}
```

Esse trecho de código dispõe as variáveis de sessão e destrói a sessão. Note que esse código também remove a configuração da variável `$action` – isso significa que inserimos a instrução `case` principal sem uma ação, desencadeando o seguinte código:

```
default:
{
   if(!check_logged_in( ))
     display_login_form($action);
   break;
}
```

Isso permitirá que outro usuário efetue logon ou que o usuário efetue logon como outra pessoa.

Implementando funções administrativas

Se alguém efetuar logon como um administrador, obterá algumas opções adicionais de menu, que podem ser vistas na Figura 30.10.

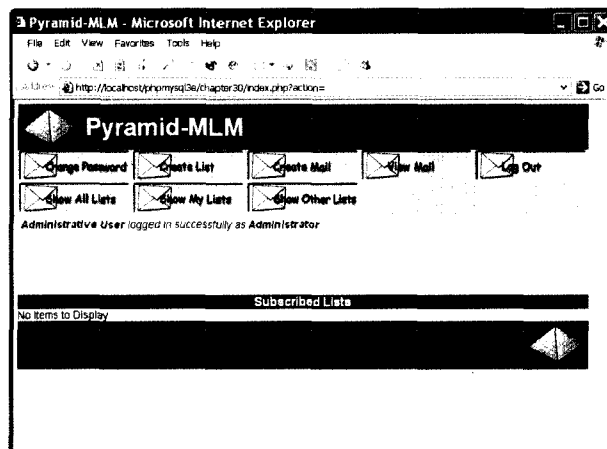



Figura 30.10 O menu de administrador permite a criação de lista de mala-direta e manutenção.

As opções extras oferecidas são: Create List (cria uma nova lista de mala-direta), Create Mail (cria um novo boletim) e View Mail (visualização e envio de boletins criados que ainda não foram enviados). Veremos cada um desses individualmente.

Criando uma nova lista

Se o administrador escolher configurar uma nova lista clicando no botão Create List, ativará a ação `create-list`, que está associada ao seguinte código:

```
case 'create-list' :
{
   display_list_form(get_email( ));
   break;
}
```

A função `display_list_form( )` exibe um formulário que permite ao administrador inserir os detalhes de uma nova lista. É possível encontrá-la na biblioteca `output_fns.php`. Simplesmente gera HTML; portanto, não iremos analisá-la aqui. A saída dessa função é mostrada na Figura 30.11.

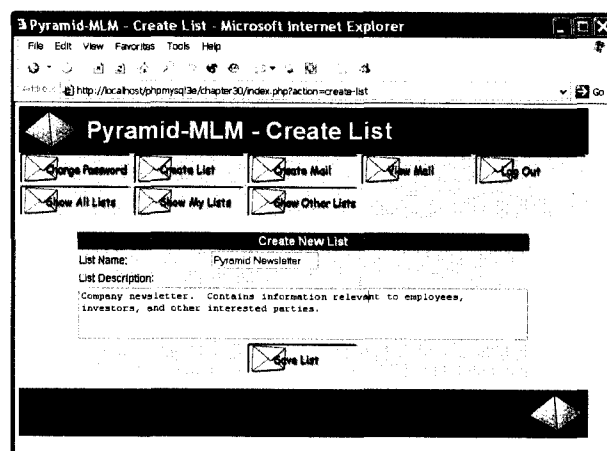



Figura 30.11 A opção Create List exige que o administrador insira um nome e uma descrição (ou blurb) para a nova lista.

Quando o administrador clicar no botão Save List, isso ativa a ação `store-list`, que desencadeia o seguinte código em `index.php`:

```
case 'store-list' :
{
  if(store_list($_SESSION['admin_user'], $_POST))
  {
    echo '<p>New list added</p><br />';
    display_items('All Lists', get_all_lists( ), 'information',
                  'show-archive','');
  }
  else
    echo '<p>List could not be stored, please try '
         .'again.</p><br /><br /><br /><br /><br />';

  break;
}
```

Como você pode ver, o código tenta armazenar os novos detalhes de lista e então exibe a nova lista de listas. Os detalhes de lista são armazenados com a função `store_list( )`. O código para essa função é mostrado na Listagem 30.14.

Listagem 30.14 **Função `store_list( )` de `mlm_fns.php` – Essa função insere uma nova lista de mala-direta no banco de dados**

```
function store_list($admin_user, $details)
{
   if(!filled_out($details))
   {
     echo 'All fields must be filled in.  Try again.<br /><br />';
     return false;
   }
  else
   {
     if(!check_admin_user($admin_user))
       return false;
       // como essa função foi chamada por alguém não-conectado como admin?

     if(!db_connect( ))
```

Listagem 30.14 **Continuação**

```
      return false;
    }

    $query = "select count(*) from lists where listname = '"
              .$details['name']."'";
    $result = mysql_query($query);
    if(mysql_result($result, 0, 0) > 0)
    {
      echo 'Sorry, there is already a list with this name.';
      return false;
    }

    $query = "insert into lists values (NULL,
                                        '".$details['name']."',
                                        '".$details['blurb']."')";

    $result = mysql_query($query);
    return $result;
  }
}
```

Essa função realiza algumas verificações de validação antes de gravar no banco de dados: ela verifica se todos os detalhes foram fornecidos, se o usuário atual é administrador e se o nome de lista é único. Se tudo correr bem, a lista é adicionada à tabela `lists` no banco de dados.

Fazendo upload de um novo boletim

Por fim, chegamos ao impulso principal dessa aplicação: fazer upload de boletins e enviá-los para listas de mala-direta.

Quando um administrador clica no botão Create Mail, ele ativa a ação `create-mail`, da seguinte maneira:

```
case 'create-mail' :
{
   display_mail_form(get_email( ));
   break;
}
```

O administrador verá o formulário mostrado na Figura 30.12.

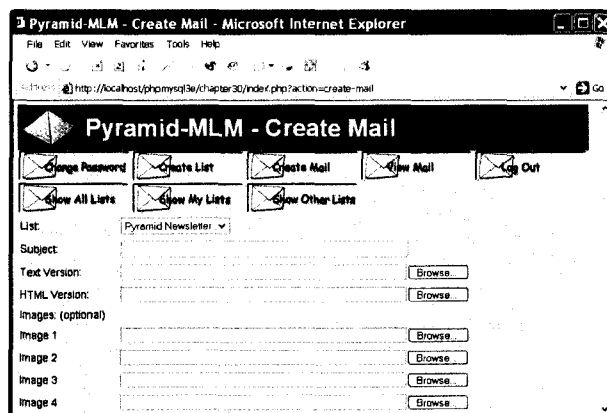



Figura 30.12 A opção Create Mail fornece ao administrador uma interface para fazer upload dos arquivos do boletim.

Lembre-se de que para essa aplicação estamos assumindo que o administrador criou um boletim off-line tanto no formato HTML como no formato de texto e fará upload das duas versões antes do envio. Escolhemos implementá-la dessa maneira para que os administradores possam utilizar seu software favorito para criar os boletins. Isso torna a aplicação mais acessível.

Você pode ver que esse formulário tem vários campos para um administrador preencher. Na parte superior está uma caixa drop-down de listas de mala-direta a partir da qual escolher. O administrador também deve preencher um assunto para o boletim – essa é a linha Subject para o e-mail final.

Todos os outros campos de formulário são campos de upload de arquivo, que você pode ver a partir dos botões Browse perto deles. A fim de enviar um boletim, um administrador deve listar tanto versões de texto como versões de HTML desse boletim (embora obviamente você pudesse alterar para atender às suas necessidades). Há também vários campos opcionais de imagem nos quais um administrador pode fazer upload de qualquer imagem que embutiu na HTML. Cada um desses arquivos deve ser especificado e carregado separadamente.

O formulário que você vê é semelhante a um formulário de upload de arquivo normal exceto que, nesse caso, nós o estamos utilizando para fazer upload de diversos arquivos. São necessárias algumas pequenas alterações na sintaxe do formulário e na maneira como lidamos com os arquivos carregados na outra extremidade.

O código para a função `display_mail_form( )` é mostrado na Listagem 30.15.

Listagem 30.15 **Função `display_mail_form( )` de `output_fns.php` – Essa função exibe o formulário de upload de arquivo**

```
function display_mail_form($email, $listid=0)
{
   // exibe formulário de html para fazer upload de uma nova mensagem
   global $table_width;
   $list = get_all_lists( );
   $lists = sizeof($list);
?>
   <table cellpadding="4" cellspacing="0" border="0"
          width="<?php echo $table_width?>">
   <form enctype='multipart/form-data' action='upload.php' method='post'>
   <tr>
     <td bgcolor="#cccccc">
       List:
     </td>
     <td bgcolor="#cccccc">
       <select name="list">
       <?php
       for($i = 0; $i<$lists; $i++)
       {
         echo '<option value = '.$list[$i][0];
         if ($listid== $list[$i][0]) echo ' selected';
        echo '>'.$list[$i][1]."</option>\n";
       }
       ?>
       </select>
     </td>
   </tr>
   <tr>
     <td bgcolor="#cccccc">
       Subject:
     </td>
     <td bgcolor="#cccccc">
       <input type="text" name="subject" value="<?php echo $subject?>"
        size = 60 ></td>
   </tr>
```

Listagem 30.15 Continuação

```
    <tr><td bgcolor="#cccccc">
      Text Version:
    </td><td bgcolor="#cccccc">
      <input type=file name='userfile[0]' size = 60>
    </td></tr>
    <tr><td bgcolor="#cccccc">
      HTML Version:
    </td><td bgcolor="#cccccc">
      <input type=file name='userfile[1]' size = 60>
    </td></tr>
    <tr><td bgcolor="#cccccc" colspan="2">Images: (optional)

<?php
    $max_images = 10;
    for($i = 0; $i<10; $i++)
    {
      echo "<tr><td bgcolor='#cccccc'>Image ". ($i+1) .' </td>';
      echo "<td bgcolor='#cccccc'>";
      echo "<input type='file' name='userfile[".($i+2)."]' size='60'></td></tr>";
    }
?>
    <tr><td colspan="2" bgcolor="#cccccc" align="center">
    <input type="hidden" name="max_images" value="<?php echo $max_images?>">
    <input type="hidden" name="listid" value="<?php echo $listid?>">
    <?php display_form_button('upload-files'); ?>
    </td>
    </form>
    </tr>
    </table>
<?php
}
```

Algo a notar aqui é que os arquivos que desejamos carregar por upload terão seus nomes inseridos em uma série de entradas, cada uma do tipo file e com nomes que variam de `userfile[0]` a `userfile[n]`. Em essência, estamos tratando esses campos de formulário da mesma maneira como trataríamos caixas de seleção e nomeando-os utilizando uma convenção de array.

Se quiser fazer upload de diversos arquivos por meio de um script de PHP, você precisa seguir essa convenção.

No script que processa esse formulário, realmente terminaremos com *três arrays*. Veremos esse script a seguir.

Tratando upload de diversos arquivos

Talvez você se lembre de que colocamos o código de upload de arquivo em um arquivo separado. A listagem completa desse arquivo, `upload.php`, é mostrada na Listagem 30.16.

Listagem 30.16 **`upload.php`** – **Esse script carrega todos os arquivos necessários para um boletim**

```
<?php
  // essa funcionalidade está em um arquivo separado para nos permitir ser
  // mais cuidadosos

  // se algo der errado, sairemos

  $max_size = 50000;
```

Listagem 30.16 **Continuação**

```
include ('include_fns.php');
session_start( );

// somente usuários admin podem fazer upload de arquivos
if(!check_admin_user( ))
{
  echo 'You do not seem to be authorized to use this page.';
  exit;
}

// configura os botões da barra de ferramentas do admin
$buttons = array( );
$buttons[0] = 'change-password';
$buttons[1] = 'create-list';
$buttons[2] = 'create-mail';
$buttons[3] = 'view-mail';
$buttons[4] = 'log-out';
$buttons[5] = 'show-all-lists';
$buttons[6] = 'show-my-lists';
$buttons[7] = 'show-other-lists';

do_html_header('Pyramid-MLM - Upload Files');

display_toolbar($buttons);

// verifica se a página está sendo chamada com os dados exigidos
if(!$_FILES['userfile']['name'][0]
   ||!$_FILES['userfile']['name'][1]
   ||!$_POST['subject']||!$_POST['list'])
{
    echo 'Problem: You did not fill out the form fully. The images are the
          only optional fields.  Each message needs a subject, text version
          and an HTML version.';
    do_html_footer( );
    exit;
}
$list = $_POST['list'];
$subject = $_POST['subject'];

if(!($conn=db_connect( )))
{
   echo '<p>Could not connect to db</p>';
   do_html_footer( );
   exit;
}

// acrescenta detalhes do correio ao bd
$query = "insert into mail values (NULL,
                                   '{$_SESSION['admin_user']}',
                                   '$subject',
                                   '$list',
                                   'STORED', NULL, NULL)";
$result = $conn->query($query);
if(!$result)
{
  do_html_footer( );
  exit;
}

//obtém o MySQL de id atribuído a esse correio
```

Listagem 30.16 Continuação

```
$mailid = $conn->insert_id;

if(!$mailid)
{
  do_html_footer( );
  exit;
}


// criação de diretório se esta não for a primeira mensagem arquivada
// isso está ok
@ mkdir('archive/'.$list, 0700);

// é um problema se a criação do diretório específico para este correio falhar
if(!mkdir('archive/'.$list."/$mailid", 0700))
{
  do_html_footer( );
  exit;
}

// itera por um array de arquivos carregados
$i = 0;
while ($_FILES['userfile']['name'][$i]&&
       $_FILES['userfile']['tmp_name'][$i]!='none')
{
  echo '<p>Uploading '.$_FILES['userfile']['name'][$i].' - ';
  echo $_FILES['userfile']['size'][$i].' bytes.</p>';
  if ($_FILES['userfile']['size'][$i]==0)
  {
    echo 'Problem: '.$_FILES['userfile']['name'][$i].
         ' is zero length';
    $i++;
    continue;
  }

  if ($_FILES['userfile']['size'][$i]>$max_size)
  {
    echo 'Problem: '.$_FILES['userfile']['name'][$i].' is over '
         .$max_size.' bytes';
    $i++;
    continue;
  }

  // gostaríamos de verificar se a imagem carregada é uma imagem
  // se getimagesize( ) pode descobrir o tamanho, provavelmente é.
  if($i>1&&!getimagesize($_FILES['userfile']['tmp_name'][$i]))
  {
    echo 'Problem: '.$_FILES['userfile']['name'][$i].
         ' is corrupt, or not a gif, jpeg or png';
    $i++;
    continue;
  }

  // file 0 (mensagem de texto) e file 1 (mensagem html) são casos especiais
  if($i==0)
    $destination = "archive/$list/$mailid/text.txt";
  else if($i == 1)
    $destination = "archive/$list/$mailid/index.html";
  else
  {
```

Listagem 30.16 Continuação

```
      $destination = "archive/$list/$mailid/"
                     .$_FILES['userfile']['name'][$i];
      $query = "insert into images
               values ($mailid,
                      '".addslashes($_FILES['userfile']['name'][$i])."',
                      '".addslashes($_FILES['userfile']['type'][$i])."'
                     )";
      $result = $conn->query($query);
    }
    //se estamos utilizando a versão PHP >= 4.03

    if (!is_uploaded_file($_FILES['userfile']['tmp_name'][$i]))
    {
      // possível ataque de upload de arquivo detectado
      echo 'Something funny happening with '
          .$_FILES['userfile']['name'].', not uploading.';
      do_html_footer( );
      exit;
    }

    move_uploaded_file($_FILES['userfile']['tmp_name'][$i],
                       $destination);
/*
    // se versão <= 4.02
    copy ($userfile[$i], $destination);

    unlink($userfile[$i]);
*/

    $i++;
  }

  display_preview_button($list, $mailid, 'preview-html');
  display_preview_button($list, $mailid, 'preview-text');
  display_button('send', "&id=$mailid");

  echo '<br /><br /><br /><br /><br />';
  do_html_footer( );
? >
```

Vamos examinar detalhadamente os passos da Listagem 30.16.

Primeiro, iniciamos uma sessão e verificamos se o usuário está conectado como um administrador – não queremos deixar qualquer outra pessoa fazer upload de arquivos.

Estritamente falando, provavelmente também deveríamos verificar as variáveis `list` e `mailid` quanto a caracteres indesejáveis, mas ignoramos isso por questão de simplicidade.

Em seguida, configuramos e enviamos os cabeçalhos para a página e validamos se o formulário foi preenchido corretamente. Isso é importante aqui uma vez que é um formulário de preenchimento muito complexo para o usuário.

Portanto, criamos uma entrada para esse correio no banco de dados e configuramos um diretório no repositório para o correio ser armazenado.

A seguir vem a parte principal do script, que verifica e move cada um dos arquivos transferidos por upload. Essa parte é diferente ao fazer upload de diversos arquivos. Agora você tem quatro arrays com que lidar; eles são chamados `$_FILES['userfile']['name']`, `$_FILES['userfile']['tmp_name']`, `$_FILES['userfile']['size']` e `$_FILES['userfile']['type']`. Eles correspondem a seus equivalentes chamados de forma semelhante em um único upload de arquivo, exceto que cada um deles é um array. O primeio arquivo no formulário será detalhado em `$_FILES['userfile']['tmp_na-`

me'][0], $_FILES['userfile']['name'][0], $_FILES['userfile']['size'][0] e $_FILES['userfile']['type'][0].

Dados esses três arrays, realizamos as verificações de segurança usuais e movemos os arquivos para dentro do repositório.

Por fim, fornecemos ao administrador alguns botões que ele pode utilizar para visualizar o boletim que fez upload antes do envio, e um botão para enviá-lo. Você pode ver a saída de upload.php na Figura 30.13.

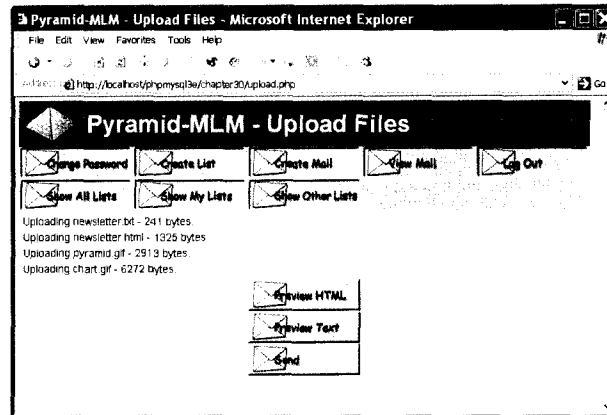



Figura 30.13 O script de upload informa os arquivos carregados e seus tamanhos.

Visualizando o boletim

Há duas maneiras de o administrador visualizar um boletim antes de enviá-lo. Pode acessar as funções de visualização a partir da tela de upload se quiser visualizar imediatamente depois de fazer upload. Ele também pode clicar no botão View Mail, que mostrará todos os boletins não enviados no sistema, se ele quiser visualizar e enviar correio mais tarde. O botão View Mail ativa a ação view-mail, que desencadeia o seguinte código:

```
case 'view-mail' :
{
   display_items('Unsent Mail', get_unsent_mail(get_email( )),
                 'preview-html', 'preview-text', 'send');
   break;
}
```

Como você pode ver, isso novamente utiliza a função display_items( ) com botões para as ações preview-html, preview-text e send.

Um ponto interessante a ser observado é que os botões "Preview" não desencadeiam realmente uma ação, mas em vez disso vinculam-se diretamente ao boletim no repositório. Se examinar outra vez as Listagens 30.7 e 30.16, você verá que utilizamos a função display_preview_button( ) para criar esses botões, em vez da função display_button( ) normal.

A função display_button( ) cria um link de imagem para um script com os parâmetros GET onde exigido; a função display_preview_button( ) fornece um link simples no repositório. Esse link se abrirá em uma nova janela com repositórios utilizando o atributo target=new da tag de âncora de HTML. Você pode ver o resultado de visualizar a versão de HTML de um boletim na Figura 30.14.

Enviando a mensagem

Clicar no botão Send de um boletim ativa a ação send, que desencadeia o seguinte código:

```
case 'send' :
{
```

```
    send($_GET['id'], $_SESSION['admin_user']);
    break;
}
```

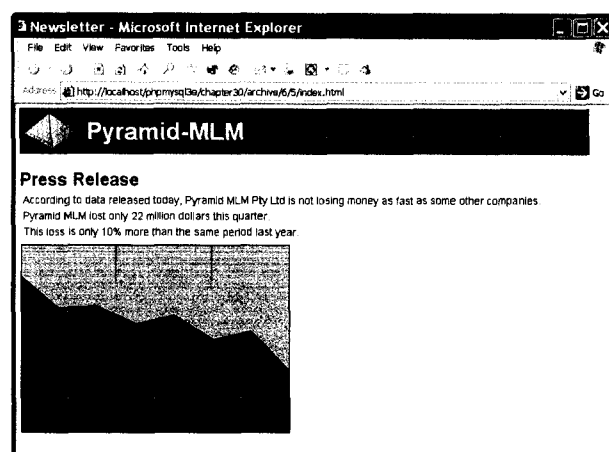



Figura 30.14 Uma visualização completa de um boletim de HTML com imagens.

Isso chama a função `send( )` que você pode encontrar na biblioteca `mlm_fns.php`. Essa é uma função bem longa. É também o ponto em que utilizamos a classe `Mail_mime`.

O código para nossa função é mostrado na Listagem 30.17.

Listagem 30.17 **Função send( ) de mlm_fns.php – Por fim essa função envia um boletim**

```
function send($mailid, $admin_user)
{
  if(!check_admin_user($admin_user))
    return false;

  if(!($info = load_mail_info($mailid)))
  {
    echo "Cannot load list information for message $mailid";
    return false;
  }
  $subject = $info['subject'];
  $listid = $info['listid'];
  $status = $info['status'];
  $sent = $info['sent'];

  $from_name = 'Pyramid MLM';

  $from_address = 'return@address';
  $query = "select email from sub_lists where listid = $listid";

  $conn = db_connect( );
  $result = $conn->query($query);
  if (!$result)
  {
    echo $query;
    return false;
  }
  else if ($result->num_rows==0)
  {
    echo "There is nobody subscribed to list number $listid";
    return false;
  }
```

Listagem 30.17 **Continuação**

```
// inclui classes de correio PEAR
require('Mail.php');
require('Mail/mime.php');

// instancia classe MIME e lhe passa o caractere de retorno de carro/quebra de linha
// caractere utilizado neste sistema
$message = new Mail_mime("\r\n");

// lê a versão de texto do boletim
$textfilename = "archive/$listid/$mailid/text.txt";
$tfp = fopen($textfilename, "r");
$text = fread($tfp, filesize($textfilename));
fclose($tfp);

// lê a versão HTML do boletim
$htmlfilename = "archive/$listid/$mailid/index.html";
$hfp = fopen($htmlfilename, "r");
$html = fread($hfp, filesize($htmlfilename));
fclose($hfp);

// acrescenta HTML e texto ao objeto mimemail
$message->setTXTBody($text);
$message->setHTMLBody($html);

// obtém a lista das imagens que têm relação com essa mensagem
$query = "select path, mimetype from images where mailid = $mailid";
$result = $conn->query($query);
if(!$result)
{
  echo '<p>Unable to get image list from database.</p>';
  return false;
}
$num = $result->num_rows;
for($i = 0; $i<$num; $i++)
{
  //carrega cada imagem a partir do disco
  $row = $result->fetch_array( );
  $imgfilename = "archive/$listid/$mailid/".$row[0];
  $imgtype = $row[1];
   // acrescenta cada imagem ao objeto
  $message->addHTMLImage($imgfilename, $imgtype, $imgfilename, true);
}

// cria corpo da mensagem
$body = $message->get( );

// cria cabeçalhos da mensagem
$from = '"'.get_real_name($admin_user).'" <'.$admin_user.'>';
$hdrarray = array('From'    => $from,
                  'Subject' => $subject);

$hdrs = $message->headers($hdrarray);

// cria o objeto real de envio
$sender =& Mail::factory('mail');

if($status == 'STORED')
{

  // envia a mensagem HTML ao administrador
```

Listagem 30.17 Continuação

```
    $sender->send($admin_user, $hdrs, $body);

    // envia a versão de texto simples da mensagem ao administrador
    mail($admin_user, $subject, $text, 'From: "'.
         get_real_name($admin_user).
         '" <'.$admin_user.">");

    echo "Mail sent to $admin_user";

    // marca o boletim como testado
    $query = "update mail set status = 'TESTED' where mailid = $mailid";
    $result = $conn->query($query);

    echo '<p>Press send again to send mail to whole list.<center>';
    display_button('send', "&id=$mailid");
    echo '</center></p>';
  }
  else if($status == 'TESTED')
  {
    //envia para a lista completa

    $query = "select subscribers.realname, sub_lists.email,
                     subscribers.mimetype
              from sub_lists, subscribers
              where listid = $listid and
                    sub_lists.email = subscribers.email";

    $result = $conn->query($query);
    if(!$result)
      echo '<p>Error getting subscriber list</p>';

    $count = 0;
    // para cada assinante
    while( $subscriber = $result->fetch_row( ))
    {
      if($subscriber[2]=='H')
      {
        //envia versão HTML para as pessoas que a quiserem
        $sender->send($subscriber[1], $hdrs, $body);
      }
      else
      {
        //envia a versão de texto para as pessoas que não querem correio HTML
        mail($subscriber[1], $subject, $text,
             'From: "'.get_real_name($admin_user).'" <'.$admin_user.">");
      }
      $count++;
    }

    $query = "update mail set status = 'SENT', sent = now( )
              where mailid = $mailid";
    $result = $conn->query($query);
    echo "<p>A total of $count messages were sent.</p>";
  }
  else if($status == 'SENT')
  {
    echo '<p>This mail has already been sent.</p>';
  }
}
```

Essa função faz várias coisas diferentes. Ela testa o correio do boletim para o administrador antes de enviá-lo. Ela prossegue monitorando o status de uma parte de correio no banco de dados. Quando o script de upload faz upload de uma parte do correio, ela configura o status inicial desse correio como "STORED".

Se a função send( ) descobre que um correio tem o status "STORED", ela atualizará esse para "TESTED" e enviará para o administrador. O status "TESTED" significa que o boletim foi enviado por e-mail para o administrador como teste. Se o status for "TESTED", ele será alterado para "SENT" e enviado para a lista inteira. Isso significa que cada correio deve ser essencialmente enviado duas vezes: uma vez no modo de teste e uma vez no modo real.

A função também envia dois tipos diferentes de e-mail: a versão texto, que envia utilizando a função mail( ) do PHP; e a versão HTML, que envia utilizando a classe Mail_mime. Utilizamos mail( ) muitas vezes neste livro, então vejamos como utilizar a classe Mail_mime. Não abordaremos essa classe de modo abrangente, mas em vez disso explicaremos como utilizá-la nessa aplicação relativamente típica. Iniciamos incluindo os arquivos de classe e criando uma instância da classe Mail_mime:

```
// inclui classes de correio PEAR
include('Mail.php');
include('Mail/mime.php');

// instancia a classe MIME e lhe passa o caractere de quebra
// de linha/retorno de carro utilizado nesse sistema
$message = new Mail_mime("\r\n");
```

Você observará que estamos incluindo dois arquivos de classe aqui. Utilizaremos a classe genérica Mail do PEAR mais tarde nesse script para realmente enviar a mensagem de correio. Essa classe acompanha sua instalação do PEAR.

A classe Mail_mime é utilizada para criar a mensagem que será enviada no formato MIME.

Em seguida, lemos as versões de texto e de HTML da mensagem de correio e as adicionamos à classe Mail_mime:

```
// lê a versão de texto do boletim
$textfilename = "archive/$listid/$mailid/text.txt";
$tfp = fopen($textfilename, "r");
$text = fread($tfp, filesize($textfilename));
fclose($tfp);

// lê a versão de HTML do boletim
$htmlfilename = "archive/$listid/$mailid/index.html";
$hfp = fopen($htmlfilename, "r");
$html = fread($hfp, filesize($htmlfilename));
fclose($hfp);

// adiciona HTML e texto ao objeto mimemail
$message->setTXTBody($text);
$message->setHTMLBody($html);
```

Portanto, carregamos os detalhes de imagem do banco de dados e fazemos loop por eles, adicionando cada imagem à parte de correio que desejamos enviar:

```
$num = $result->num_rows;
for($i = 0; $i<$num; $i++)
{
  //carrega cada imagem do disco
  $row = $result->fetch_array( );
  $imgfilename = "archive/$listid/$mailid/".$row[0];
  $imgtype = $row[1];
   // adiciona cada imagem ao objeto
  $message->addHTMLImage($imgfilename, $imgtype, $imgfilename, true);
}
```

Os parâmetros que passamos para addHTMLImage( ) são o nome do arquivo da imagem (ou também poderíamos passar os dados da imagem), o tipo MIME da imagem, o nome do arquivo novamente e true para indicar que o primeiro parâmetro é um nome de arquivo em vez dos dados do arquivo. (Se quiséssemos passar dados brutos da imagem, passaríamos os dados, o tipo MIME, um parâmetro vazio e false.) Esses parâmetros são um pouco complicados.

Nessa etapa, precisamos criar o corpo da mensagem. Precisamos fazer isso antes que possamos configurar os cabeçalhos da mensagem. Criamos o corpo como a seguir:

```
// cria o corpo da mensagem
$body = $message->get( );
```

Em seguida, podemos criar os cabeçalhos da mensagem com uma chamada para a função headers( ) de Mail_mime:

```
// cria os cabeçalhos da mensagem
$from = '"'.get_real_name($admin_user).'" <'.$admin_user.'>';
$hdrarray = array(
              'From'    => $from,
              'Subject' => $subject);
$hdrs = $message->headers($hdrarray);
```

Por fim, tendo configurado a mensagem, podemos enviá-la. Para tanto, precisamos instanciar a classe Mail do PEAR e passar-lhe a mensagem que criamos. Iniciamos instanciando a classe, como a seguir:

```
// cria o objeto real de envio
$sender =& Mail::factory('mail');
```

(Aqui, o parâmetro 'mail' apenas informa à classe Mail para utilizar a função mail( ) para enviar mensagens. Também poderíamos utilizar 'sendmail' ou 'smtp' como o valor desse parâmetro com os resultados óbvios.)

Em seguida, enviamos o correio para cada um dos nossos assinantes. Fazemos isso recuperando e fazendo loop por cada um dos usuários que fizeram assinatura dessa lista e utilizando Mail send( ) ou mail( ) normal dependendo da preferência de tipo MIME do usuário:

```
if($subscriber[2]=='H')
{
   //envia versão de HTML às pessoas interessadas
   $sender->send($subscriber[1], $hdrs, $body);
}
else
{
   //envia versão de texto às pessoas que não querem correio de HTML
   mail($subscriber[1], $subject, $text,
                 'From: "'.get_real_name($admin_user).'" <'.$admin_user.">");
}
```

O primeiro parâmetro de $sender->send( ) deve ser o endereço de e-mail do usuário; o segundo, os cabeçalhos; e o terceiro, o corpo da mensagem.

É isso aí! Concluímos agora a construção da aplicação de lista de mala-direta.

Estendendo o projeto

Como de costume com esses projetos, há muitas maneiras de estender a funcionalidade. Talvez você queira:

- Confirmar o quadro de assinantes de modo que as pessoas não possam fazer uma assinatura sem sua permissão. Isso em geral é feito enviando e-mail para suas contas e excluindo aqueles que não respondem. Essa abordagem também limpa qualquer endereço de e-mail incorreto do banco de dados.

- Fornecer ao administrador poderes de aprovar ou rejeitar usuários que desejam fazer assinatura de suas listas.
- Adicionar funcionalidade de lista aberta que permite a qualquer membro enviar e-mail para a lista.
- Permitir que apenas os membros registrados vejam o repositório para uma lista de mala-direta particular.
- Permitir que os usuários procurem listas que correspondam aos critérios específicos. Por exemplo, talvez os usuários estejam interessados em boletins de golfe. Uma vez que o número de boletins cresce além de um tamanho particular, uma pesquisa seria útil para localizar os específicos.
- Torne o programa mais eficiente para lidar com listas grandes de mala-direta. Para fazer isso, use um gerenciador de listas de mala-direta construído para esse propósito, como o exmlm, que pode enfileirar e enviar mensagens de uma forma multithread. Chamar `mail( )` várias vezes não é muito eficiente no PHP, fazendo um back end do PHP inadequado para grandes listas de assinantes. Naturalmente, você ainda poderia construir o front-end no PHP, mas empregue o exmlm para realizar o trabalho maçante.

A seguir

No próximo capítulo, implementaremos uma aplicação de fórum Web que permitirá aos usuários terem discussões on-line estruturadas por tópico e threads conversacionais.

31

Construindo fóruns Web

UMA BOA MANEIRA DE CONSEGUIR QUE OS USUÁRIOS retornem para seu site é oferecer fóruns. Os fóruns podem ser utilizados para propósitos tão variados como grupos de discussão filosófica e suporte técnico de produto. Neste capítulo, implementamos um fórum Web no PHP. Uma alternativa é utilizar um pacote existente, como o Phorum, para configurar seus fóruns.

Os fóruns Web também são às vezes chamados de grupos de discussão ou *grupos de discussão encadeada*. A idéia de um fórum é que pessoas podem enviar artigos ou perguntas e outros podem ler e responder a suas perguntas. Cada tópico de discussão em um fórum é chamado de *thread*, ou encadeamento .

Para este projeto, implementaremos um fórum Web chamado blah-blah. Os usuários serão capazes de:

- Iniciar novos encadeamentos de discussão para enviar artigos;
- Enviar artigos em resposta a artigos existentes;
- Visualizar artigos que foram enviados;
- Visualizar os encadeamentos de conversa no fórum;
- Visualizar o relacionamento entre artigos, isto é, ver quais artigos são respostas para outros;

O problema

Configurar um fórum é realmente um problema bastante interessante. Precisaremos de algumas maneiras de armazenar os artigos em um banco de dados com autores, títulos, datas e as informações de conteúdos. À primeira vista, isso talvez não pareça muito diferente do banco de dados Book-O-Rama.

Entretanto, a maneira como a maior parte do software de discussão encadeada funciona é que, além de mostrar a você os artigos disponíveis, ele mostrará o relacionamento entre artigos. Isto é, você é capaz de ver quais artigos são respostas a outros (e de qual artigo eles são uma seqüência) e quais artigos são novos tópicos de discussão.

Você pode ver exemplos de fóruns de discussão que implementam isso em muitos lugares, incluindo Slashdot:

http://slashdot.org

Decidir como exibir esses relacionamentos exigirá pensar cuidadosamente. Para esse sistema, um usuário deve ser capaz de visualizar uma mensagem individual, um encadeamento de discussão com os relacionamentos mostrados ou todos os encadeamentos no sistema. Os usuários também devem ser capazes de postar novos tópicos ou respostas. Essa é a parte fácil.

Componentes da solução

Como dissemos anteriormente, armazenar e recuperar o autor e o texto de uma mensagem é fácil. A parte mais difícil dessa aplicação é encontrar uma estrutura de banco de dados que armazenará as informações que queremos e uma maneira de navegar por essa estrutura eficientemente.

A estrutura de artigos em uma discussão poderia se parecer com a mostrada na Figura 31.1.

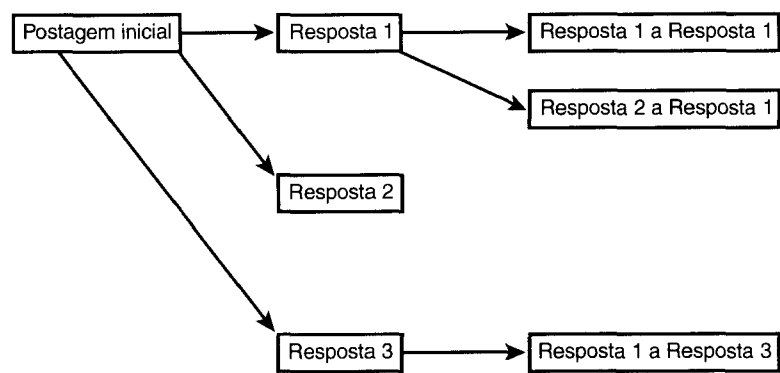



Figura 31.1 Um artigo em uma discussão encadeada poderia ser o primeiro artigo em um novo tópico, mas mais comumente é uma resposta a outro artigo.

Nesse diagrama, você pode ver que temos uma postagem inicial iniciando um tópico, com três respostas. Algumas respostas têm respostas. Essas respostas poderiam ter respostas e assim por diante.

Examinar o diagrama fornece um indício sobre como podemos armazenar e recuperar os dados de artigo e os links entre artigos. Esse diagrama mostra uma *estrutura de árvore*. Se fez muita programação, você saberá que essa é uma das estruturas de dados básicas utilizadas. No diagrama há *nós* (ou artigos) e *links* (ou relacionamentos) entre artigos – assim como em qualquer estrutura de árvore. Se você não estiver familiarizado com árvores como uma estrutura de dados, não se preocupe – abordaremos os princípios básicos à medida que avançarmos.)

Os truques para fazer tudo isso funcionar são:

1. Encontrar uma maneira de mapear essa estrutura de árvore no armazenamento – em nosso caso, dentro de um banco de dados do MySQL.
2. Encontrar um modo de reconstruir os dados como exigido.

Iniciaremos implementando um banco de dados do MySQL que permitirá armazenar artigos durante sua utilização. Construiremos interfaces simples para permitir salvar artigos.

Quando carregarmos a lista de artigos para visualizar, carregaremos os cabeçalhos de cada artigo em uma classe de PHP `tree_node`. Cada `tree_node` conterá cabeçalhos de um artigo e um conjunto das respostas para esse artigo.

As respostas serão armazenadas em um array. Cada resposta será por si própria um `tree_node`, que pode conter um array de respostas para esse artigo, que são em si próprias `tree_nodes` e assim por diante. Isso continua até que alcançamos os chamados *nós-folha* da árvore, os nós que não têm nenhuma resposta. Portanto, teremos uma estrutura de árvore que é semelhante à Figura 31.1.

Um pouco de terminologia: a mensagem a que estamos respondendo pode ser chamada de nó *pai* do nó atual. Todas as respostas à mensagem podem ser chamadas *filhos* do nó atual. Se você imaginar que essa estrutura de árvore é igual a uma árvore genealógica, isso será fácil de lembrar.

O primeiro artigo nessa estrutura de árvore – o artigo sem pai – é às vezes chamado nó-*raiz*.

Nota Chamar o primeiro artigo de raiz pode não ser intuitivo porque normalmente desenhamos o nó-raiz na parte superior dos diagramas, ao contrário das raízes de árvores verdadeiras.

Para construir e exibir essa estrutura de árvore, escreveremos funções recursivas. (Discutimos recursão no Capítulo 5.)

Decidimos utilizar uma classe para essa estrutura porque é a maneira mais fácil de construir uma estrutura de dados complexa, em expansão dinâmica para essa aplicação. Também significa que temos código elegante e bem simples para fazer algo bem complexo.

Visão geral da solução

Para entender realmente o que fizemos com esse projeto, provavelmente é uma boa idéia percorrer passo a passo o código, o que faremos em breve. Há menos volume nessa aplicação que em alguns outros, mas o código é um pouco mais complexo.

Há somente três páginas reais na aplicação. Teremos uma página principal de índice que mostra todos os artigos no fórum como links para os artigos. A partir daqui, você será capaz de adicionar um novo artigo, visualizar um artigo listado ou alterar a maneira como os artigos são visualizados expandindo e recolhendo ramos da árvore. (Informações adicionais sobre isso em um minuto.) A partir da visualização de artigo, você será capaz de postar uma resposta a esse artigo ou visualizar as respostas existentes para esse artigo. A nova página de artigo permite inserir uma nova postagem, quer seja uma resposta a uma mensagem existente, quer seja uma nova mensagem não-relacionada.

O diagrama de fluxo de sistema é mostrado na Figura 31.2.

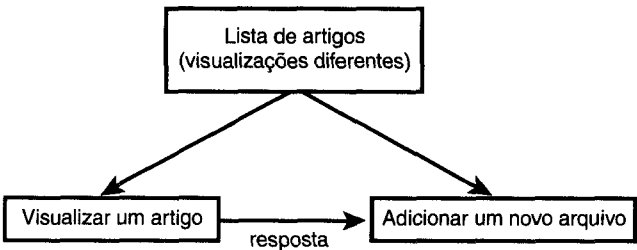



Figura 31.2 Há três partes principais do sistema de fórum blah-blah.

Um resumo dos arquivos nessa aplicação é mostrado na Tabela 31.1.

Tabela 31.1 **Arquivos na aplicação de fórum Web**

Nome	Tipo	Descrição
`index.php`	Aplicação	A página principal que os usuários verão quando entrarem no site. Contém uma lista desmontável e expansível de todos os artigos no site.
`new_post.php`	Aplicação	Formulário utilizado para postar novos artigos.
`store_new_post.php`	Aplicação	Armazena artigos inseridos no formulário `new_post.php`.
`view_post.php`	Aplicação	Exibe uma mensagem individual e uma lista das respostas para essa mensagem.
`treenode_class.php`	Biblioteca	Contém a classe `treenode`, que utilizaremos para exibir a hierarquia de mensagens.
`include_fns.php`	Biblioteca	Junta todas as outras bibliotecas de funções para essa aplicação (os outros arquivos de tipo Library listados aqui).
`data_valid_fns.php`	Biblioteca	Funções de validação de dados.
`db_fns.php`	Biblioteca	Funções de conectividade de banco de dados.

Tabela 31.1 **Continuação**

Nome	Tipo	Descrição
discussion_fns.php	Biblioteca	Funções para lidar com armazenamento e recuperação de mensagens.
output_fns.php	Biblioteca	Funções para gerar saída de HTML.
create_database.sql	SQL	SQL para configurar o banco de dados exigido para essa aplicação.

Vamos em frente e vejamos a implementação.

Projetando o banco de dados

Há alguns atributos que precisaremos armazenar sobre cada artigo postado para o fórum: a pessoa que o escreveu, chamada *poster*; o título do artigo; quando ele foi postado; e o corpo de artigo. Portanto, precisaremos de uma tabela de artigos. Criaremos um ID único para cada artigo, chamado `postid`.

Cada artigo precisa ter algumas informações sobre onde ele pertence na hierarquia. Poderíamos armazenar as informações sobre os filhos de um artigo com o artigo. Entretanto, cada artigo pode ter muitas respostas, e isso pode levar a alguns problemas na construção de banco de dados. Como cada artigo só pode ser uma resposta a um outro, é mais fácil armazenar uma referência ao artigo pai, isto é, o artigo a que este artigo está respondendo.

Isso fornece os seguintes dados a armazenar para cada artigo:

- `postid`: um ID único para cada artigo.
- `parent`: o `postid` do artigo pai.
- `poster`: o autor desse artigo.
- `title`: o título desse artigo.
- `posted`: a data e a hora que o artigo foi postado.
- `message`: o corpo do artigo.

Adicionaremos duas otimizações a isso.

Quando estivermos tentando determinar se um artigo tem alguma resposta, teremos de executar uma consulta para ver se algum outro artigo tem este artigo como pai. Precisaremos dessas informações para cada mensagem (ou postagem) que listamos. Quanto menos consultas tivermos de executar, mais rápido nosso código executará. Podemos remover a necessidade para essas consultas adicionando um campo para mostrar se há alguma resposta. Chamaremos esse campo de `children` e o tornaremos efetivamente um booleano – o valor será `1` se o nó tiver filhos e `0` se não tiver.

Você sempre paga um preço pelas otimizações. Aqui estamos escolhendo armazenar dados redundantes. Como estamos armazenando os dados de duas maneiras, devemos ter cuidado para nos certificar de que as duas representações estão de acordo uma com a outra. Quando adicionamos filhos, devemos atualizar o pai. Se permitirmos a exclusão dos filhos, precisamos atualizar o nó pai para nos certificar de que o banco de dados está consistente. Nesse projeto não construiremos um recurso para excluir artigos, então evitaremos metade desse problema. Se você decidir estender esse código, tenha essa questão em mente.

Faremos uma outra otimização: separaremos os corpos de mensagem dos outros dados e os armazenaremos em uma tabela separada. A razão para isso é que esse atributo terá o tipo do MySQL `text`. Ter esse tipo em uma tabela torna as consultas lentas. Como faremos muitas consultas pequenas para construir a estrutura de árvore, isso a tornaria bastante lenta. Com os corpos de mensagem

em uma tabela separada, podemos simplesmente recuperá-los quando um usuário quiser ver uma mensagem particular.

O MySQL pode pesquisar registros de tamanho fixo mais rápido que os registros de tamanho variável. Se precisarmos utilizar os dados de tamanho variável, podemos ajudar criando índices nos campos que serão utilizados para pesquisar o banco de dados. Para alguns projetos, seria mais útil deixar o campo de texto no mesmo registro que tudo mais e especificar índices em todas as colunas em que pesquisaremos. Mas os índices levam tempo para gerar, e os dados em nossos fóruns provavelmente serão alterados o tempo todo, então precisaríamos gerar novamente nossos índices com freqüência.

Além disso, adicionaremos um atributo area no caso de decidirmos mais tarde implementar vários *chats*, ou bate-papos, com a aplicação. Não implementaremos isso aqui, mas dessa maneira ele está reservado para utilização futura.

Dadas todas essas considerações, o SQL para criar o banco de dados para o banco de dados de fórum é mostrado na Listagem 31.1.

Listagem 31.1 **create_database.sql – SQL para criar o banco de dados de discussão**

```
create database discussion;

use discussion;

create table header
(
  parent int not null,
  poster char(20) not null,
  title char(20) not null,
  children int default 0 not null,
  area int default 1 not null,
  posted datetime not null,
  postid int unsigned not null auto_increment primary key
);

create table body
(
  postid int unsigned not null primary key,
  message text
);


grant select, insert, update, delete
on discussion.*
to discussion@localhost identified by 'password';
```

Você pode criar essa estrutura do banco de dados executando esse script por meio do MySQL da seguinte maneira:

```
mysql -u root -p < create_database.sql
```

Você precisará fornecer sua senha de root. Além disso, você deve provavelmente alterar a senha que configuramos para o usuário do fórum de discussão para algo melhor.

Para entender como essa estrutura armazenará os artigos e o relacionamento entre eles, veja a Figura 31.3.

Como você pode ver, o campo pai para cada artigo no banco de dados armazena o postid do artigo acima dele na árvore. O artigo pai é o artigo que está sendo respondido.

Você também pode ver que o nó-raiz, postid 1, não tem nenhum pai. Todos os novos tópicos de discussão estarão nessa posição. Para artigos desse tipo, armazenamos seu pai como um 0 (zero) no banco de dados.

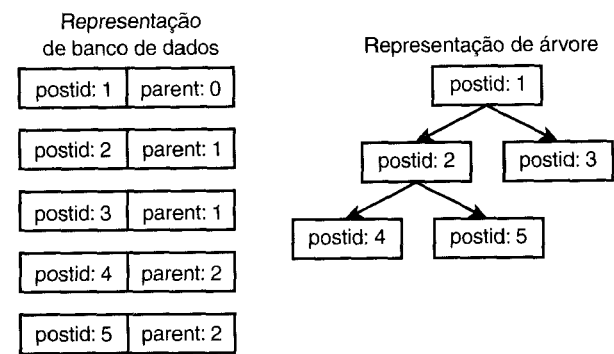



Figura 31.3 O banco de dados possui a estrutura de árvore de uma forma relacional achatada.

Visualizando a árvore de artigos

Em seguida, precisamos de uma maneira de expor informações do banco de dados e representá-las de volta na estrutura de árvore. Faremos isso com a página principal, `index.php`. Para propósitos dessa explicação, inserimos algumas postagens de exemplo por meio dos scripts de postagem de artigo `new_post.php` e `store_new_post.php`. Veremos esses na próxima seção.

Abordaremos a lista de artigo primeiro porque ela é a espinha dorsal do site. Depois disso, tudo será fácil.

A Figura 31.4 mostra a visualização inicial dos artigos no site que um usuário veria.

Essa figura mostra todos os artigos iniciais. Nenhum desses são respostas; todos eles são o primeiro artigo sobre um tópico particular.

Nesse caso, você verá que temos várias opções. Há uma barra de menu que permitirá adicionar uma nova postagem e expandir ou recolher a visualização que temos dos artigos.

Para entender o que isso significa, veja as postagens. Algumas têm símbolos de adição ao lado delas. Isso significa que esses artigos foram respondidos. Para ver as respostas de um artigo particular, você pode clicar no símbolo de adição. O resultado de clicar em um desses símbolos é mostrado na Figura 31.5.

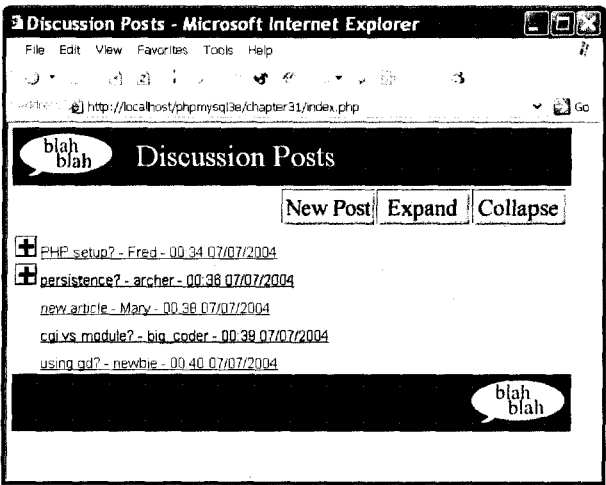



Figura 31.4 A visualização inicial da lista de artigo mostra os artigos na forma "recolhida".

Como você pode ver, clicar no símbolo de adição exibiu as respostas desse primeiro artigo. O símbolo de adição agora virou um símbolo de subtração. Se clicarmos nesse símbolo, todos os artigos nesse encadeamento serão recolhidos, retornando a visualização inicial.

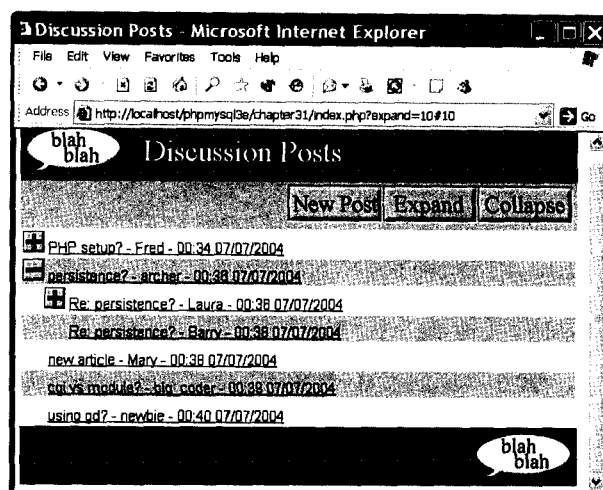



Figura 31.5 O encadeamento de discussão sobre persistência foi expandido.

Talvez você também note que uma das respostas tem um símbolo de adição ao lado dela. Isso significa que há respostas para essa resposta. Isso pode continuar até uma profundidade arbitrária e você pode visualizar cada conjunto de resposta clicando no símbolo de adição apropriado.

As duas opções de barra de menu, Expand e Collapse, expandirão e recolherão todos os possíveis encadeamentos, respectivamente. O resultado de clicar no botão Expand é mostrado na Figura 31.6.

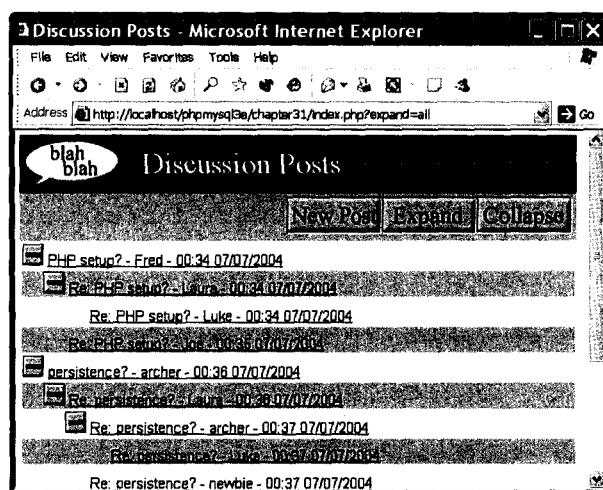



Figura 31.6 Agora todos os encadeamentos foram expandidos.

Se examinar cuidadosamente as Figuras 31.5 e 31.6, você pode ver que estamos passando alguns parâmetros de volta para `index.php` na linha de comando. Na Figura 31.5, o URL é semelhante a:

```
http://localhost/phpmysql3e/chapter31/index.php?expand=10#10
```

O script lê isso como "Expanda o item com `postid` 10". O # é simplesmente uma âncora de HTML que rolará a página para baixo até a parte que acabou de ser expandida.

Na Figura 31.6, o URL é:

```
http://localhost/phpmysql3e/chapter31/index.php?expand=all
```

Ao clicar no botão Expand, o parâmetro `expand` é passado com o valor `all`.

Expandindo e recolhendo

Vejamos como isso é feito examinando o script `index.php`, mostrado na Listagem 31.2.

Listagem 31.2 **`index.php` – Script para criar a visualização de artigo na página principal da aplicação**

```
<?php
 include ('include_fns.php');
 session_start( );

 // verifica se criamos nossa variável de sessão
 if(!isset($_SESSION['expanded']))
 {
   $_SESSION['expanded'] = array( );
 }

 // verifica se um botão expand foi pressionado
 // expand pode ser igual a 'all' ou um postid ou pode não estar configurado
 if(isset($_GET['expand']))
 {
   if($_GET['expand'] == 'all')
     expand_all($_SESSION['expanded']);
   else
     $_SESSION['expanded'][$_GET['expand']] = true;
 }

 // verifica se um botão collapse foi pressionado
 // collapse pode ser igual a all ou a um postid ou pode não estar configurado
 if(isset($_GET['collapse']))
 {
   if($_GET['collapse']=='all')
     $_SESSION['expanded'] = array( );
   else
     unset($_SESSION['expanded'][$_GET['collapse']]);
 }

 do_html_header('Discussion Posts');

 display_index_toolbar( );

 // exibe a visualização em árvore das conversas
 display_tree($_SESSION['expanded']);

 do_html_footer( );
?>
```

Esse script utiliza três variáveis para fazer seu trabalho. Essas variáveis são:

- A variável de sessão `expanded`, que monitora os encadeamentos a serem expandidos. Isso pode ser mantido de visualização para visualização, assim podemos ter diversos encadeamentos expandidos. Essa variável é um array associativo que contém o `postid` de artigos a terem suas respostas exibidas expandidas.
- O parâmetro `expand`, que instrui o script sobre os novos encadeamentos que devem ser expandidos.
- O parâmetro `collapse`, que instrui o script sobre os encadeamentos que devem ser recolhidos.

Quando clicamos em um símbolo de adição ou de subtração ou no botão Expand ou Collapse, eles chamarão outra vez o script index.php com os novos parâmetros para expand ou collapse. Utilizamos expanded de página a página para monitorar os encadeamentos que devem ser expandidos em qualquer dada visualização.

O script index.php começa iniciando uma sessão e adicionando a variável expanded como uma variável de sessão se isso já não foi feito. Depois disso, o script verifica se foi passado a ele um parâmetro expand ou collapse e modifica o array expanded adequadamente. Veja o código para o parâmetro expand:

```
if(isset($_GET['expand']))
{
 if($_GET['expand'] == 'all')
   expand_all($_SESSION['expanded']);
 else
   $_SESSION['expanded'][$_GET['expand']] = true;
}
```

Se clicarmos no botão Expand, a função expand_all( ) será chamada para adicionar todos os encadeamentos que têm respostas no array expandido. (Veremos isso em um momento.)

Se estivermos tentando expandir um encadeamento particular, nos será passado um postid via expand. Portanto, adicionamos uma nova entrada ao array expanded para refletir isso.

A função expand_all( ) é mostrada na Listagem 31.3.

Listagem 31.3 **Função expand_all( ) de discussion_fns.php – Processa o array $expanded para expandir todos os encadeamentos no fórum**

```
function expand_all(&$expanded)
{
 // marca todos os encadeamentos com filhos para serem motrados no modo expandido
 $conn = db_connect( );
 $query = 'select postid from header where children = 1';
 $result = $conn->query($query);
 $num = $result->num_rows;
 for($i = 0; $i<$num; $i++)
 {
   $this_row = $result->fetch_row( );
   $expanded[$this_row[0]]=true;
 }
}
```

Essa função executa uma consulta de banco de dados para descobrir os encadeamentos no fórum que têm respostas, da seguinte maneira:

```
select postid from header where children = 1
```

Todos os artigos retornados são então adicionados ao array expanded. Executamos essa consulta para economizar tempo mais tarde. Poderíamos simplesmente adicionar todos os artigos à lista expandida, mas seria desperdício tentar processar respostas que não existem.

Recolher os artigos funciona de maneira semelhante mas oposta, assim:

```
if(isset($_GET['collapse']))
{
 if($_GET['collapse']=='all')
   $_SESSION['expanded'] = array( );
 else
   unset($_SESSION['expanded'][$_GET['collapse']]);
}
```

Você pode remover itens do array expanded removendo sua configuração. Removemos o encadeamento que será recolhido ou removemos a configuração do array inteiro se a página inteira tiver de ser recolhida.

Tudo isso é pré-processado, então sabemos quais artigos devem ser exibidos e quais não devem. A parte-chave do script é a chamada a display_tree($_SESSION['expanded']); que realmente gerará a árvore de artigos exibidos.

Exibindo os artigos

Vejamos a função display_tree( ), mostrada na Listagem 31.4.

Listagem 31.4 **Função display_tree( ) de output_fns.php – Cria o nó-raiz da estrutura de árvore**

```
function display_tree($expanded, $row = 0, $start = 0)
{
  // exibe a visualização de árvore de conversas

  global $table_width;
  echo "<table width=\"$table_width\">";

  // vê se estamos exibindo a lista inteira ou uma sublista
  if($start>0)
    $sublist = true;
  else
    $sublist = false;

  // constrói estrutura de árvore para representar o resumo de conversa
  $tree = new treenode($start, '', '', '', 1, true, -1, $expanded, $sublist);

  // diz para a árvore exibir a si própria
  $tree->display($row, $sublist);

  echo '</table>';
}
```

O principal papel dessa função é criar o nó-raiz da estrutura de árvore. Nós a utilizamos tanto para exibir o índice inteiro como para criar subárvores de respostas na página view_post.php. Como você pode ver, ela recebe três parâmetros. O primeiro, $expanded, é a lista de postids dos artigos a exibir de forma expandida. O segundo, $row, é um indicador do número de linha que será utilizado para elaborar as cores alternadas das linhas na lista.

O terceiro parâmetro, $start, informa a função onde começar a exibir os artigos. Esse é o postid do nó-raiz para a árvore que será criada e exibida. Se estivermos exibindo tudo, como estamos na página principal, isso será 0 (zero), o que significa exibir todos os artigos sem pai. Se esse parâmetro for 0, configuramos $sublist como false e exibimos toda a árvore.

Se o parâmetro for maior que 0, o utilizamos como o nó-raiz da árvore para exibir, configuramos $sublist como true e construímos e exibimos apenas parte da árvore. (Utilizaremos isso no script view_post.php.)

A coisa mais importante que essa função faz é instanciar uma instância da classe treenode que representa a raiz da árvore. Isso não é realmente um artigo mas atua como o pai de todos os artigos de primeiro nível, que não têm nenhum pai. Depois que a árvore foi construída, simplesmente chamamos sua função de exibição para realmente exibir a lista de artigos.

Utilizando a classe treenode

O código para a classe treenode é mostrado na Listagem 31.5. (Talvez você ache útil nessa etapa consultar novamente o Capítulo 6, para se lembrar como as classes funcionam.)

Listagem 31.5 Classe **treenode** de **treenode_class.php** – A espinha dorsal da aplicação

```
<?php
// funções para carregar, construir e
// exibir a árvore estão neste arquivo

class treenode
{
 // cada nó na árvore tem variáveis de membro contendo
 // todos os dados para um postagem exceto o corpo da mensagem
 public $m_postid;
 public $m_title;
 public $m_poster;
 public $m_posted;
 public $m_children;
 public $m_childlist;
 public $m_depth;

 public function __construct($postid, $title, $poster, $posted, $children,
                  $expand, $depth, $expanded, $sublist)
 {
   // o construtor configura as variáveis de membro, mas mais
   // importante recursivamente cria partes inferiores da árvore
   $this->m_postid = $postid;
   $this->m_title = $title;
   $this->m_poster = $poster;
   $this->m_posted = $posted;
   $this->m_children =$children;
   $this->m_childlist = array( );
   $this->m_depth = $depth;

   // só nos importamos com o que está abaixo deste nó se ele
   // tiver filhos e estiver marcado para ser expandido
   // sublistas são sempre expandidas
   if(($sublist||$expand) && $children)
   {
     $conn = db_connect( );

     $query = "select * from header where parent = $postid order by posted";
     $result = $conn->query($query);

     for ($count=0; $row = @$result->fetch_assoc( ); $count++)
     {
       if($sublist||$expanded[ $row['postid'] ] == true)
         $expand = true;
       else
         $expand = false;
       $this->m_childlist[$count]= new treenode($row['postid'],$row['title'],
                                  $row['poster'],$row['posted'],
                                  $row['children'], $expand,
                                  $depth+1, $expanded, $sublist);
     }
   }
 }
```

Listagem 31.5 **Continuação**

```
function display($row, $sublist = false)
{
  // como é um objeto, é responsável por se mostrar

  // $row nos diz de qual linha da exibição estamos acima
  // então sabemos de que cor deve ser

  // $sublist nos diz se estamos na página principal
  // ou na página de mensagem. As páginas de mensagem devem ter
  // $sublist = true.
  // Em uma sublista, todas as mensagens são expandidas e não existem
  // símbolos de "+" ou "-".

  // se este é o nó raiz vazio pular exibição
  if($this->m_depth > -1)
  {
    //cor alterna linhas
    echo '<tr><td bgcolor = ';
    if ($row%2)
      echo "'#cccccc'>";
    else
      echo "'#ffffff'>";

    // recua respostas até a profundidade do aninhamento
    for($i = 0; $i < $this->m_depth; $i++)
    {
      echo "<img src = 'images/spacer.gif' height = '22'
                  width = '22' alt = '' valign = 'bottom' />";
    }

    // exibe + ou - ou um espaçador
    if ( !$sublist && $this->m_children && sizeof($this->m_childlist))
    // estamos na página principal, temos alguns filhos e eles estão expandidos
    {
      // estamos expandidos - oferece botão para recolher
      echo "<a href = 'index.php?collapse=".
                 $this->m_postid."#$this->m_postid'
           ><img src = 'images/minus.gif' valign = 'bottom'
           height = '22' width = '22' alt = 'Collapse Thread' border = '0'
           /></a>";
    }
    else if(!$sublist && $this->m_children)
    {
      // estamos recolhidos - oferece botão para expandir
      echo "<a href = 'index.php?expand=".
           $this->m_postid."#$this->m_postid'><img src = 'images/plus.gif'
           height = '22' width = '22' alt = 'Expand Thread' border = '0'
           /></a>";
    }
    else
    {
      // não temos filhos, ou estamos na sublista, não oferece botão
      echo "<img src = 'images/spacer.gif' height = '22' width = '22'
              alt = '' valign = 'bottom' />";
    }

    echo " <a name = $this->m_postid ><a href =
           'view_post.php?postid=$this->m_postid'>$this->m_title -
           $this->>m_poster - ".reformat_date($this->m_posted).'</a>';
    echo '</td></tr>';
```

Listagem 31.5 Continuação

```
      // incrementa contador de linha para alternar cores
      $row++;
    }
    // chama display em cada um dos filhos deste nó
    // note que um nó só terá filhos em sua lista se expandido
    $num_children = sizeof($this->m_childlist);
    for($i = 0; $i < $num_children; $i++)
    {
      $row = $this->m_childlist[$i]->display($row, $sublist);
    }
    return $row;
  }

}

?>
```

Essa classe contém a funcionalidade que guia a visualização de árvore nessa aplicação.

Uma instância da classe `treenode` contém detalhes sobre uma única postagem e vincula-se a todos as postagens de resposta dessa classe. Isso fornece as seguintes variáveis membro:

```
public $m_postid;
public $m_title;
public $m_poster;
public $m_posted;
public $m_children;
public $m_childlist;
public $m_depth;
```

Note que o `treenode` não contém o corpo do artigo. Não há nenhuma necessidade de carregar isso até que um usuário vá ao script `view_post.php`. Precisamos tentar fazer isso relativamente rápido, uma vez que estamos fazendo uma grande quantidade de manipulação de dados para exibir a lista de árvore e precisamos recalcular quando a página é atualizada ou um botão é pressionado.

Essas variáveis seguem um esquema de atribuição de nomes comumente utilizado em aplicações orientadas a objetos – iniciar variáveis com `m_` para lembrar que elas são variáveis membro da classe.

A maioria dessas variáveis corresponde diretamente a linhas da tabela `header` em nosso banco de dados. As exceções são `$m_childlist` e `$m_depth`. Utilizaremos a variável `$m_childlist` para armazenar as respostas para esse artigo. A variável `$m_depth` armazenará o número de níveis de árvore abaixo do que estamos – isso será utilizado para criar a exibição.

A função construtora configura os valores de todas as variáveis, da seguinte maneira:

```
Public function __construct($postid, $title, $poster, $posted, $children,
                            $expand, $depth, $expanded, $sublist)
{
  // o construtor configura as variáveis membro, mas o que é mais
  // importante é ele que cria recursivamente as partes inferiores da árvore
  $this->m_postid = $postid;
  $this->m_title = $title;
  $this->m_poster = $poster;
  $this->m_posted = $posted;
  $this->m_children =$children;
  $this->m_childlist = array( );
  $this->m_depth = $depth;
```

Quando construímos o `treenode` de raiz a partir de `display_tree( )` a partir da página principal, na verdade estamos criando um nó fictício sem nenhum artigo associado a ele. Passamos alguns valores iniciais:

```
$tree = new treenode($start, '', '', '', 1, true, -1, $expanded, $sublist);
```

Isso cria um nó-raiz com um `$postid` de zero. Isso pode ser utilizado para localizar todas as postagens de primeiro nível porque elas têm um pai de zero. Configuramos a profundidade como -1 porque esse nó não é realmente parte da exibição. Todas as postagens de primeiro nível terão uma profundidade de zero e estão na parte esquerda da tela. As profundidades subseqüentes avançam para a direita.

O mais importante que acontece nesse construtor é que os nós filhos desse nó são instanciados. Começamos esse processo verificando se é necessário expandir os nós filhos. Só realizamos esse processo se um nó tiver alguns filhos e se escolhermos exibi-los:

```
if(($sublist||$expand) && $children)
{
  $conn = db_connect( );
```

Então, nos conectamos ao banco de dados e recuperamos todas as postagens filhas, da seguinte maneira:

```
  $query = "select * from header where parent = $postid order by posted";
  $result = $conn>query($query);
```

Depois, preenchemos o array `$m_childlist` com instâncias da classe `treenode`, contendo a resposta para a postagem armazenada nesse `treenode`, como a seguir:

```
for ($count=0; $row = @mysql_fetch_array($result); $count++)
{
  if($sublist||$expanded[ $row['postid'] ] == true)
    $expand = true;
  else
    $expand = false;
  $this->m_childlist[$count]= new treenode($row['postid'],$row['title'],
                              $row['poster'],$row['posted'],
                              $row['children'], $expand,
                              $depth+1, $expanded, $sublist);
}
```

Essa última linha criará os novos `treenodes`, seguindo exatamente o mesmo processo que acabamos de percorrer passo a passo, mas para o próximo nível abaixo na árvore. Essa é a parte recursiva. Um nó de árvore pai está chamando o construtor `treenode`, passando seu próprio `postid` como pai e adicionando um a sua própria profundidade antes de passá-lo.

Cada `treenode` por sua vez será criado e criará seus próprios filhos até esgotarmos as respostas ou níveis a que queremos expandir.

Depois que tudo isso for feito, chamamos a função de exibição da raiz `treenode` (essa está de novo em `display_tree( )`), da seguinte maneira:

```
$tree->display($row, $sublist);
```

A função `display( )` inicia verificando se isso é o nó-raiz fictício:

```
if($this->m_depth>-1)
```

Dessa maneira, o nó fictício pode ser deixado fora da exibição. No entanto, não queremos pular completamente o nó-raiz. Não queremos que ele apareça, mas ele precisa notificar seus filhos de que eles precisam se exibir.

A função `display( )` então inicia desenhando a tabela contendo os artigos. Ela utiliza o operador de módulo (%) para decidir a cor de fundo que essa linha deve ter (de modo que elas sejam alternadas):

```
//pinta as linhas com cores alternadas
echo '<tr><td bgcolor = ';
```

```
if ($row%2)
  echo "'#cccccc'>";
else
  echo "'#ffffff'>";
```

Então, utiliza a variável `$m_depth` de membro para descobrir o quanto recuar o item atual. Você verá, examinando outra vez as figuras, que quanto mais profundo o nível em que uma resposta está, mais ainda ela será recuada. Isso é feito da seguinte maneira:

```
// recua respostas até a profundidade do aninhamento
for($i = 0; $i<$this->m_depth; $i++)
{
  echo '<img src="images/spacer.gif" height="22"
                  width="22" alt="" valign="bottom">';
}
```

A próxima parte da função decide se fornece um botão de adição ou de subtração ou simplesmente nada:

```
// exibe + ou - ou um espaçador
if ( !$sublist && $this->m_children && sizeof($this->m_childlist))
// estamos na página principal, tem algumas filhos e eles são expandidos
{
  // estamos expandidos - oferece botão para recolher
       echo '<a href="index.php?collapse='.
            $this->m_postid.'#'.$this->m_postid.'"
      ><img src="images/minus.gif" valign="bottom"
      height="22" width="22" alt="Collapse Thread" border="0"></a>';
}
else if(!$sublist && $this->m_children)
{
  // estamos recolhidos - oferece botão para expandir
  echo '<a href = "index.php?expand='.
      $this->m_postid.'#'.$this->m_postid.'"><img src="images/plus.gif"
      height="22" width="22" alt="Expand Thread" border="0"></a>';
}
else
{
  // não temos nenhum filho ou estamos em uma sublista, não recebemos botão
  echo '<img src="images/spacer.gif" height="22" width="22"
           alt="" valign="bottom">';
}
```

Em seguida, exibimos os detalhes reais desse nó:

```
echo " <a name = $this->m_postid ><a href =
           'view_post.php?postid=$this->m_postid'>$this->m_title -
            $this->m_poster - ".reformat_date($this->m_posted).'</a>';
echo '</td></tr>';
```

Alteramos a cor para a próxima linha:

```
// incrementa contador de linha para alternar cores
$row++;
```

Depois disso, há algum código que será executado por todos os `treenodes`, incluindo o da raiz, da seguinte maneira:

```
// chama display sobre todos os filhos desse nó
// observe que um nó somente terá filhos em sua lista se expandido
$num_children = sizeof($this->m_childlist);
for($i = 0; $i<$num_children; $i++)
{
```

```
    $row = $this->m_childlist[$i]->display($row, $sublist);
  }
  return $row;
```

Novamente essa é uma chamada de função recursiva, que chama cada um dos filhos desse nó para exibirem a si próprios. Passamos a esses filhos a cor atual da linha e fazemos com que eles a passem de volta quando terminarem de pintar, então podemos monitorar a alternação de cores.

Isso é tudo para essa classe. O código é relativamente complexo. Você poderia querer experimentar a execução da aplicação e então voltar a examiná-lo novamente quando se sentir familiarizado com o que ele faz.

Visualizando artigos individuais

A chamada `display_tree( )` termina oferecendo links para um conjunto de artigos. Se clicarmos em um desses artigos, iremos para o script `view_post.php`, com um parâmetro do `postid` do artigo a ser visualizado. A saída de exemplo desse script é mostrada na Figura 31.7.

Esse script mostra o corpo de mensagem, bem como as respostas para essa mensagem. Você verá que as respostas são novamente exibidas como uma árvore, mas completamente expandidas dessa vez e sem qualquer botão de adição ou subtração. Esse é o efeito da opção `$sublist` entrando em ação. Vejamos o código para `view_post.php`, mostrado na Listagem 31.6.

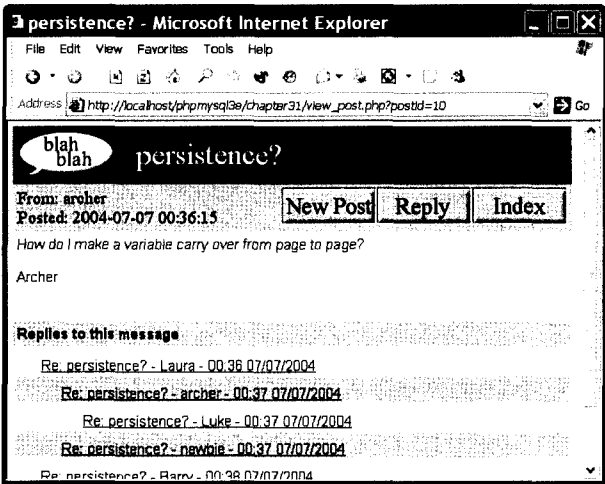



Figura 31.7 Agora podemos ver o corpo de mensagem para essa postagem.

Listagem 31.6 **view_post.php – Exibe o corpo de uma única mensagem**

```
<?php
  // inclui bibliotecas de funções
  include ('include_fns.php');
  $postid = $_GET['postid'];
  // obtém detalhes da postagem
  $post = get_post($postid);

  do_html_header($post['title']);

  // exibe a postagem
  display_post($post);

  // se a postagem tiver quaisquer respostas, mostra a visualização de árvore delas
  if($post['children'])
  {
```

Listagem 31.6 **Continuação**

```
    echo '<br /><br />';
    display_replies_line( );
    display_tree($_SESSION['expanded'], 0, $postid);
  }

  do_html_footer( );
?>
```

Esse script utiliza três chamadas de função principais para fazer seu trabalho: `get_post( )`, `display_post( )` e `display_tree( )`.

A função `get_post( )` extrai os detalhes de função do banco de dados. O código para essa função é mostrado na Listagem 31.7.

Listagem 31.7 **Função `get_post( )` de `discussion_fns.php` – Recupera uma mensagem do banco de dados**

```
function get_post($postid)
{
  // extrai uma postagem do banco de dados e retorna como um array

  if(!$postid) return false;

  $conn = db_connect( );

  //obtém todas as informações de cabeçalho a partir de 'header'
  $query = "select * from header where postid = $postid";
  $result = $conn_query($query);
  if($result->num_rows($result)!=1)
    return false;
  $post = $result->fetch_assoc( );

  // obtém mensagem a partir do corpo e a adiciona ao resultado anterior
  $query = "select * from body where postid = $postid";
  $result2 = $conn->query($query);
  if($result2->num_rows>0)
  {
    $body = $result2->fetch_assoc( );
    if($body)
   {
      $post['message'] = $body['message'];
    }
  }
  return $post;
}
```

Quando essa função receber um `postid`, realizará as duas consultas requeridas para recuperar o cabeçalho e o corpo dessa postagem e os juntará em um único array associativo que ela então retorna.

Os resultados dessa função `get_post( )` então são passados para a função `display_post( )` a partir de `output_fns.php`. Isso simplesmente imprime o array com alguma formatação de HTML; portanto, não o incluímos aqui.

Por fim, o script `view_post.php` verifica se há alguma resposta para este artigo e chama `display_tree( )` para mostrar o formato da sublista – isto é, completamente expandida sem sinais de adição ou subtração.

Adicionando novos artigos

Depois disso tudo, vejamos agora como uma nova postagem é adicionada ao fórum. Um usuário pode fazer isso de duas maneiras: primeiro, clicando no botão New Post na página de índice e, segundo, clicando no botão Reply na página `view_post.php`.

Essas duas ações ativam o mesmo script, `new_post.php`, apenas com parâmetros diferentes. A Figura 31.8 mostra a saída de `new_post.php` quando o alcançamos pressionando o botão Reply.

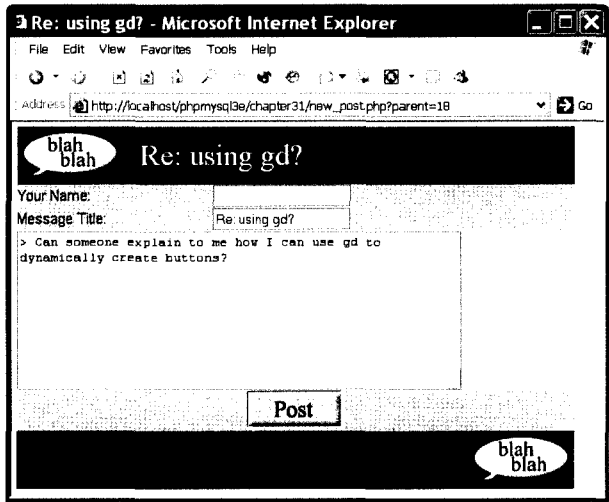



Figura 31.8 As respostas têm o texto do original automaticamente inserido e marcado.

Primeiro, examine o URL mostrado na Figura 31.8.

```
http://localhost/phpmysql3e/chapter31/new_post.php?parent=18
```

O parâmetro passado como `parent` será o `postid` pai da nova postagem. Se você clicar em New Post em vez de clicar em Reply, obterá `parent=0` no URL.

Segundo, você verá que para uma resposta, o texto da mensagem original é inserido e marcado com um caractere > como é o caso na maioria dos programas de leitura de correio e de notícia.

Terceiro, você pode ver que o título dessa mensagem assume por padrão o título da mensagem original prefixado com `Re:`.

Vejamos o código que produz essa saída. Esse código é mostrado na Listagem 31.8.

Listagem 31.8 **`new_post.php` – Permite a um usuário digitar uma nova postagem ou responder a uma postagem existente**

```
<?php
include ('include_fns.php');

$title = $_POST['title'];
$poster = $_POST['poster'];
$message = $_POST['message'];

if(isset($_GET['parent']))
  $parent = $_GET['parent'];
else
  $parent = $_POST['parent'];


if(!$area)
  $area = 1;
```

Listagem 31.8 Continuação

```
if(!$error)
{
  if(!$parent)
  {
    $parent = 0;
    if(!$title)
      $title = 'New Post';
  }
  else
  {
    // obtém o nome da postagem
    $title = get_post_title($parent);

    // append Re:
    if(strstr($title, 'Re: ') == false )
      $title = 'Re: '.$title;

    //certifica-se de que o título ainda caberá no bd
    $title = substr($title, 0, 20);

    //prefixa um padrão de citação para a postagem a que você está respondendo
    $message = add_quoting(get_post_message($parent));
  }
}
do_html_header($title);

display_new_post_form($parent, $area, $title, $message, $poster);

if($error)
{
   echo 'Your message was not stored.
         Make sure you have filled in all fields and try again.';
}

do_html_footer( );
?>
```

Depois de alguma configuração inicial, esse script verifica se o pai é zero ou outra coisa. Se for zero, esse é um novo tópico, e mais trabalho é necessário.

Se essa for uma resposta (`$parent` é o `postid` de um artigo existente), então o script irá continuar e configurar o título e o texto da mensagem original, da seguinte maneira:

```
// obtém o nome da postagem
$title = get_post_title($parent);

// acrescenta Re:
if(strstr($title, 'Re: ') == false )
  $title = 'Re: '.$title;

//certifica-se de que o título ainda caberá no bd
$title = substr($title, 0, 20);

//prefixa um padrão de citação para a postagem a que você está respondendo
$message = add_quoting(get_post_message($parent));
```

As funções que ele usa aqui são `get_post_title( )`, `get_post_message( )` e `add_quoting( )`. Essas funções são todas da biblioteca `discussion_fns.php`. Elas são mostradas nas Listagens 31.9, 31.10 e 31.11, respectivamente.

Listagem 31.9 **Função `get_post_title( )` de `discussion_fns.php` – Recupera o título de uma mensagem do banco de dados**

```
function get_post_title($postid)
{
  // extrai o nome de uma postagem a partir do banco de dados

  if(!$postid) return '';

  $conn = db_connect( );

  //obtém todas as informações de cabeçalho a partir de 'header'
  $query = "select title from header where postid = $postid";
  $result = $conn->query($query);
  if($result->num_rows!=1)
    return '';
 $this_row = $result->fetch_array( );
  return  this_row[0];

}
```

Listagem 31.10 **Função `get_post_message( )` de `discussion_fns.php` – Recupera o corpo de uma mensagem do banco de dados**

```
function get_post_message($postid)
{
  // extrai a mensagem de uma postagem a partir do banco de dados

  if(!$postid) return '';

  $conn = db_connect( );

  $query = "select message from body where postid = $postid";
  result = $conn->query($query);
  if($result->num_rows>0)
 {
   $this_row = $result->fetch_array( );
  return  this_row[0];
  }
}
```

Essas duas primeiras funções recuperam o cabeçalho e o corpo de um artigo (respectivamente) a partir do banco de dados.

Listagem 31.11 **Função `add_quoting( )` de `discussion_fns.php` – Recua um texto de mensagem com símbolos >**

```
function add_quoting($string, $pattern = '> ')
{
  // adiciona um padrão de citação para marcar texto citado em sua resposta
  return $pattern.str_replace("\n", "\n$pattern", $string);
}
```

A função `add_quoting( )` reformata a string para iniciar cada linha do texto original com um símbolo, que assume o padrão de >.

Depois que o usuário digita sua resposta e clica no botão Post, ele será levado ao script `store_new_post.php`. A saída de exemplo desse script é mostrada na Figura 31.9.

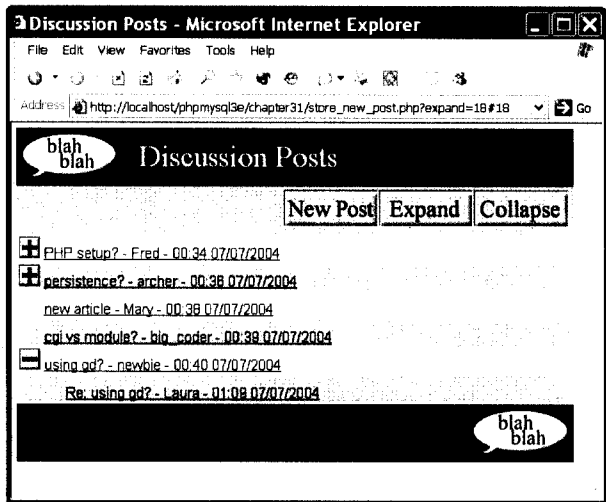



Figura 31.9 A nova postagem está agora visível na árvore.

A nova postagem é mostrada na Figura 31.9, sob `Re: using gd? - Laura - 01:09 07/07/2004`. Exceto por isso, essa página é semelhante à página `index.php` convencional.

Vejamos o código para `store_new_post.php`. Esse código é mostrado na Listagem 31.12.

Listagem 31.12 **`store_new_post.php`** – **Coloca a nova postagem no banco de dados**

```
<?php
  include ('include_fns.php');
  if($id = store_new_post($_POST))
  {
    include ('index.php');
  }
  else
  {
    $error = true;
    include ('new_post.php');
  }

?>
```

Como você pode ver, esse é um script curto. Sua principal tarefa é chamar a função `store_new_post( )`. Essa página não tem nenhum conteúdo visual próprio. Se o armazenamento for bem-sucedido, vemos a página de índice. Caso contrário, voltamos para a página `new_post.php`, para que o usuário possa tentar novamente. A função `store_new_post( )` é mostrada na Listagem 31.13.

Listagem 31.13 **Função `store_new_post( )` de `discussion_fns.php`** – **Valida e armazena a nova postagem no banco de dados**

```
function store_new_post($post)
{
  // valida a limpeza e armazena uma nova postagem

  $conn = db_connect( );
  // verifica se nenhum campo está em branco
  if(!filled_out($post))
  {
    return false;
  }
  $post = clean_all($post);
```

Listagem 31.13 **Continuação**

```
//verifica se existe um pai
if($post['parent']!=0)
{
  $query = "select postid from header where postid = '".$post['parent']."'";
  $result = mysql_query($query);
  if(mysql_numrows($result)!=1)
  {
    return false;
  }
}

// verifica se não há uma duplicata
$query = "select header.postid from header, body where
          header.postid = body.postid and
          header.parent = ".$post['parent']." and
          header.poster = '".$post['poster']."' and
          header.title = '".$post['title']."' and
          header.area = ".$post['area']." and
          body.message = '".$post['message']."'";
$result = mysql_query($query);
if (!$result)
{
   return false;
}
if($result->num_rows>0)
{
 $this_row = $result->fetch_aray( );
 return $this_row[0]
}
$query = "insert into header values
          ('".$post['parent']."',
           '".$post['poster']."',
           '".$post['title']."',
           0,
           '".$post['area']."',
           now( ),
           NULL
          )";
$result = $conn->query($query);
if (!$result)
{
   return false;
}
// observe que nosso pai agora tem um filho
$query = 'update header set children = 1 where postid = '.$post['parent'];
$result = $conn->query($query);
if (!$result)
{
   return false;
}

// localiza o id da nossa postagem, observe que poderiam ser vários cabeçalhos
// que são idênticos exceto pelo id e provavelmente data/hora da postagem
$query = "select header.postid from header left join body
          on header.postid = body.postid
                 where parent = '".$post['parent']."'
                 and poster = '".$post['poster']."'
                 and title = '".$post['title']."'
                 and body.postid is NULL";
$result = $conn->query($query);
if (!$result)
```

Listagem 31.13 Continuação

```
    {
      return false;
    }
    if($result->num_rows>0)
   {
    $this_row = @result->fetch_array( );
     $id = $this_row[0];

   if($id)
   {
      $query = "insert into body values ($id, '".$post['message']."')";
      $result = $conn->query($query);
      if (!$result)
      {
        return false;
      }

     return $id;
   }

}
```

Essa é uma função longa, mas não é muito complexa. Essa função é longa somente porque inserir uma postagem significa inserir entradas nas tabelas de corpo e cabeçalho e atualizar a linha do artigo do pai na tabela de cabeçalhos para mostrar que agora ele tem filhos.

Esse é o final do código para a aplicação do fórum Web.

Adicionando extensões

Há muitas extensões que poderiam ser adicionadas a este projeto:

- Você poderia adicionar navegação às opções de visualização, para que a partir de uma postagem você pudesse navegar para a próxima mensagem, a mensagem anterior, a mensagem logo depois do encadeamento ou a mensagem anterior no encadeamento.
- Você poderia adicionar uma interface de administração para configurar novos fóruns e excluir postagens antigas.
- Você poderia adicionar autenticação de usuário de modo que somente usuários registrados pudessem postar.
- Você poderia adicionar algum tipo de mecanismo de moderação ou censura.

Vejamos sistemas existentes para idéias.

Utilizando um sistema existente

Um sistema existente dignos de nota é o Phorum, um projeto de fóruns Web Open Source (de código-fonte aberto). Ele tem navegação e semântica diferentes do nosso, mas sua estrutura é relativamente fácil e personalizada para se ajustar ao seu próprio site. Um recurso notável do Phorum é que ele pode ser configurado pelo usuário real para exibir tanto em uma visualização simples como em uma visualização encadeada. Você pode descobrir mais sobre ele em:
http://www.phorum.org

A seguir

No Capítulo 32, utilizaremos o formato PDF para fornecer documentos atraentes, que podem ser impressos consistentemente e oferecem uma relativa proteção contra adulteração. Isso é útil para uma série de aplicações baseadas em serviço, como gerar contratos on-line.

32

Gerando documentos personalizados no formato PDF (Portable Document Format)

NOS SITES BASEADOS EM SERVIÇO, às vezes precisamos entregar documentos personalizados, gerados em resposta à entrada de nossos visitantes. Isso pode ser utilizado para fornecer um formulário preenchido automaticamente ou gerar documentos personalizados, como documentos legais, cartas ou certificados.

Nosso exemplo neste capítulo apresentará a um usuário uma habilidosa página de avaliação on-line e gerará um certificado.

Explicaremos:

- Como utilizar processamento de string do PHP para integrar um modelo com os dados de um usuário para criar um documento Rich Text Format (RTF);
- Como utilizar uma abordagem semelhante para gerar um documento Portable Document Format (PDF);
- Como utilizar funções PDFlib do PHP para gerar um documento PDF semelhante.

O problema

Neste projeto, daremos aos nossos visitantes um exame consistindo em várias perguntas. Se eles responderem corretamente a perguntas suficientes, geraremos um certificado para eles a fim de mostrar que passaram no exame.

Para que um computador possa marcá-las facilmente, nossas perguntas serão de múltipla escolha, consistindo em uma pergunta e um número de respostas potenciais. Somente uma das respostas potenciais para cada pergunta será correta.

Se um usuário alcançar um nível de aprovação nas perguntas, ele receberá um certificado.

O formato de arquivo ideal para nosso certificado deve:

1. Ser fácil de projetar.
2. Ser capaz de conter uma variedade de elementos diferentes como mapa de bits e imagens vetoriais.
3. Resultar em uma impressão de alta qualidade.
4. Exigir somente que um pequeno arquivo seja carregado.
5. Ser gerado quase instantaneamente.

6. Ter um custo baixo de produção.
7. Trabalhar em muitos sistemas operacionais.
8. Ser difícil de duplicar ou modificar fraudulentamente.
9. Não exigir nenhum software especial para visualização ou impressão.
10. Exibir e imprimir consistentemente para todos os destinatários.

Como muitas decisões que precisamos tomar de vez em quando, provavelmente precisaremos nos comprometer ao escolher um formato de entrega para atender à maior quantidade possível desses dez atributos.

Avaliando formatos de documento

A decisão mais importante que precisamos tomar é quanto aos formatos em que devemos entregar o certificado. As opções incluem papel, texto em ASCII, HTML, Microsoft Word ou outros formatos do processador de texto, Rich Text Format, PostScript e Portable Document Format. Considerando os dez atributos listados anteriormente, podemos considerar e comparar algumas de nossas opções.

Papel

Entregar o certificado em papel tem algumas vantagens óbvias. Temos controle completo sobre o processo. Podemos ver exatamente como será gerada a saída de cada certificado antes de enviá-lo para o destinatário. Não precisamos nos preocupar com software ou largura de banda e o certificado poderia ser impresso com medidas antifalsificação.

Ele atenderia a todas as nossas necessidades exceto pelos atributos 5 e 6. O certificado não poderia ser criado e entregue rapidamente. A entrega postal poderia levar dias ou semanas dependendo de nossa localização e da do destinatário.

A impressão e a postagem de cada certificado também custariam de alguns centavos a alguns dólares e provavelmente custaria mais em manipulação. A entrega eletrônica automática seria mais barata.

ASCII

Entregar documentos como ASCII ou texto simples tem algumas vantagens. A compatibilidade não será nenhum problema. A largura de banda requerida seria pequena, então o custo seria muito baixo. A simplicidade do resultado final facilitará muito projetar e gerar rapidamente um script.

Se apresentarmos um arquivo ASCII a nossos visitantes, porém, temos muito pouco controle sobre a aparência do certificado. Não podemos controlar fontes ou quebras de página. Podemos somente incluir texto e ter muito pouco controle sobre formatação. Não temos controle sobre a duplicação ou modificação do documento de um destinatário. Esse é o método que mais facilita para o destinatário alterar fraudulentamente seu certificado.

HTML

Uma escolha óbvia para entregar um documento na Web é HTML. A Hypertext Markup Language é projetada especificamente para esse propósito. Como sem dúvida você já está a par, a HTML inclui controle de formatação, sintaxe para incluir objetos como imagens e é compatível (com alguma variação) com uma variedade de sistemas operacionais e software. A HTML é relativamente simples; portanto, será fácil desenhar a página e um script poderá rapidamente gerar e entregar essa página.

As desvantagens da utilização de HTML para essa aplicação incluem: suporte limitado para impressão de formatos relacionados como quebras de página, pouca consistência da saída em platafor-

mas e programas diferentes e qualidade variável de impressão. Além disso, embora a HTML possa incluir qualquer tipo de elemento externo, a capacidade de o navegador exibir ou utilizar esses elementos não pode ser garantida para fontes incomuns.

Formatos do processador de texto

Particularmente para projetos de intranet, fornecer documentos como documentos de processador de texto faz algum sentido. Entretanto, para um projeto de Internet, utilizar um formato patenteado de processador de texto excluirá alguns visitantes, mas dado seu domínio do mercado, o Microsoft Word faria sentido. A maioria dos usuários terá acesso ao Word ou a um processador de texto que tentará ler arquivos Word.

Os usuários Windows sem Word podem fazer download do freeware Word Viewer em:

http://www.microsoft.com/office/000/viewers.asp

Gerar um documento como um documento Microsoft Word tem algumas vantagens. Contanto que você tenha uma cópia do Word, projetar um documento é fácil. Temos muito bom controle sobre a aparência impressa de nossos documentos e bastante flexibilidade com seu conteúdo. Você também pode tornar relativamente difícil para o destinatário modificar o documento instruindo o Word a pedir uma senha.

Infelizmente, arquivos do Word podem ser grandes, particularmente se contiverem imagens ou outros elementos complexos. Além disso, não há nenhuma maneira fácil de gerá-los dinamicamente com o PHP. O formato é documentado, mas é um formato binário e a documentação do formato é condicionada a uma licença de uso. É possível gerar documentos do Word com um objeto COM, mas definitivamente isso não é simples.

Outra nova possibilidade que você agora pode considerar é o OpenOffice Writer, que tem a dupla vantagem de não ser um software patenteado e de não utilizar um formato de arquivo baseado em XML.

O Word 2003 agora também suporta nativamente um formato de arquivo XML. É possível fazer download da DTD(Document Type Definition) para Word e outros produtos do Office em Microsoft.com. Procure por "Office 2003 XML Reference Schemas". Essa seria uma opção válida, mas não simples.

Rich Text Format

O Rich Text Format ou RTF fornece a maior parte do poder do Word, mas os arquivos são mais fáceis de gerar. Ainda temos a flexibilidade sobre o layout e a formatação da página impressa. Podemos ainda incluir elementos como imagens vetoriais ou imagens de mapa de bits. Além disso, podemos estar relativamente seguros de que o usuário verá um resultado semelhante ao nosso quando visualizarem ou imprimirem o documento.

O RTF é um formato de texto do Microsoft Word. Ele foi concebido como um formato de troca para transferir documentos entre diferentes programas. De certa maneira, ele é semelhante à HTML. Utiliza sintaxe e palavras-chave em vez de dados binários para transportar informações de formatação. Ele é, portanto, relativamente legível por humanos.

O formato está bem documentado. A especificação está livremente disponível e pode ser encontrada em:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp

A maneira mais fácil de gerar um documento RTF é escolher uma opção Save As RTF no seu processador de texto. Como os arquivos de RTF contêm somente texto, é possível gerá-los diretamente e os já existentes podem ser facilmente modificados.

Como é documentado e livremente disponível, o formato RTF é legível por mais softwares do que o formato binário do Word. Apesar disso, esteja ciente de que usuários que abrem um arquivo complexo de RTF em versões mais antigas do Word ou processadores de texto diferentes freqüentemente verão resultados um pouco diferentes. Cada nova versão do Word introduz novas pala-

vras-chave no RTF, de modo que implementações mais antigas normalmente ignorarão controles que elas não entendem ou escolheram não implementar.

A partir de nossa lista original, seria fácil projetar um certificado de RTF utilizando Word ou outro processador de texto; ele poderia conter uma variedade de elementos diferentes como imagens vetoriais e de mapa de bits; fornecer uma impressão de alta qualidade; ser gerado fácil e rapidamente; e ser entregue eletronicamente a baixo custo.

Ele funcionará com uma variedade de aplicações e sistemas operacionais, embora com resultados um pouco variáveis. No lado negativo, um documento de RTF pode ser fácil e livremente modificado por qualquer pessoa, o que é um problema para um certificado e alguns outros tipos de documento. O tamanho do arquivo talvez se torne moderadamente grande para documentos complexos.

O RTF é uma boa opção para muitas aplicações de entrega de documento, então nós o utilizaremos como uma opção aqui.

PostScript

O PostScript, da Adobe, é uma linguagem de descrição de página (page description language – PDL). O PostScript é uma linguagem de programação complexa e poderosa projetada para representar documentos de maneira independente de dispositivo – isto é, uma descrição que produzirá resultados consistentes entre dispositivos diferentes como impressoras e monitores. Ele está muito bem documentado. Pelo menos três livros de extensão completa estão disponíveis, bem como incontáveis Web sites.

Um documento de Postscript pode conter formatação muito precisa, texto, imagens, fontes embutidas e outros elementos. Você pode facilmente gerar um documento Postscript a partir de uma aplicação imprimindo-o em um driver de impressora PostScript. Se estiver interessado, você pode até aprender a programá-lo diretamente.

Documentos PostScript são bem portáveis. Eles fornecem impressões consistentes de alta qualidade a partir de diferentes dispositivos e sistemas operacionais diferentes.

Há dois lados negativos significativos em utilizar PostScript para distribuir documentos:

- Os arquivos podem ser enormes.
- Muitas pessoas precisarão fazer download de software adicional para utilizá-los.

A maioria dos usuários de Unix será capaz de lidar com arquivos Postscript, mas os usuários Windows normalmente precisarão fazer download de um visualizador como o GSview, que utiliza o interpretador Ghostscript Postscript. Esse software está disponível para uma ampla variedade de plataformas. Embora ele seja disponível livremente, não queremos forçar as pessoas a fazer download de mais software.

Você pode ler mais sobre Ghostscript em:

http://www.ghostscript.com/

e fazer download em:

http://www.cs.wisc.edu/~ghost/

Para nossa aplicação atual, o PostScript está muito bem cotado para saída consistente de alta qualidade, mas não alcança a maioria das nossas outras necessidades.

Portable Document Format

Felizmente, há um formato com a maior parte das capacidades do PostScript, mas com vantagens significativas. O Portable Document Format (também da Adobe) foi projetado como uma maneira de distribuir documentos que se comportariam de maneira consistente em plataformas diferentes e entregaria saída previsível e de alta qualidade em monitores ou em papel.

A Adobe descreve PDF como "o padrão aberto *de facto* para distribuição eletrônica de documentos em todo o mundo. O Adobe PDF é um formato universal de arquivo que preserva todas as

fontes, formatação, cores e imagens gráficas de qualquer documento de origem, independente da aplicação e da plataforma utilizadas para criá-lo. Os arquivos PDF são compactos e podem ser compartilhados, visualizados, navegados e impressos exatamente da maneira desejada por qualquer pessoa com um Adobe Acrobat Reader gratuito".

O PDF é um formato aberto e a documentação está disponível em

http://partners.adobe.com/asn/tech/pdf/specifications.jsp

assim como em muitos outros Web sites e um livro oficial.

Avaliando os resultados em relação aos atributos que desejamos, o PDF parece muito bom: os documentos PDF oferecem uma saída consistente e de alta qualidade, são capazes de conter elementos como imagens de mapas de bit e imagens vetoriais, podem utilizar compactação para criar um arquivo pequeno, podem ser entregues eletronicamente e com baixo custo, são utilizáveis nos sistemas operacionais mais importantes e incluem controles de segurança.

Opondo-se ao uso do PDF está o fato de que a maioria dos programas utilizados para criar documentos PDF é comercial.

Um leitor é requerido para visualizar arquivos PDF, mas o Acrobat Reader é disponibilizado gratuitamente pela Adobe para Windows, Unix e Macintosh. Muitos visitantes do seu site já conhecerão a extensão .pdf e muitos provavelmente já terão o leitor instalado.

Os arquivos PDF são uma boa maneira de distribuir documentos imprimíveis e atraentes, particularmente os que você não deseja que os destinatários possam modificar com facilidade. Veremos duas maneiras diferentes de gerar um certificado PDF.

Componentes da solução

Para fazer o sistema funcionar, precisaremos ser capazes de examinar o conhecimento dos usuários e (supondo que eles passem no teste) gerar um certificado que informe o desempenho do usuário. Experimentaremos gerar esse certificado de três maneiras diferentes: duas utilizando o PDF e uma utilizando o RTF.

Vamos ver os requisitos de cada um desses componentes em detalhes.

Sistema de pergunta e resposta

Fornecer um sistema flexível para avaliação on-line que permitisse uma variedade de tipos diferentes de pergunta, vários tipos de mídia para suportar informações, feedback útil sobre respostas erradas e coleta e relatório inteligentes de estatísticas, seria uma tarefa complexa por si só.

Neste capítulo, estamos principalmente interessados no desafio de gerar documentos personalizados para entrega pela Web, então construiremos somente um sistema muito simples de questionário. O questionário não conta com nenhum software especial. Ele utiliza um formulário HTML para fazer perguntas e um script de PHP para processar as respostas. Temos feito isso desde o Capítulo 1.

Software de geração de documentos

Nenhum software adicional é necessário no servidor Web para gerar documentos RTF ou PDF a partir de modelos, mas você precisará de software para criar os modelos. Para utilizar as funções de criação PDF do PHP, você precisará ter compilado suporte de PDF no PHP. (Discutiremos mais sobre isso em um minuto.)

Software para criar modelo RTF

Você pode utilizar o processador de texto de sua escolha para gerar arquivos de RTF. Utilizamos Microsoft Word para criar nosso modelo de certificado. O modelo de certificado está incluído no CD-ROM no diretório `Chapter 32`.

Se preferir outro processador de texto, ainda seria uma boa idéia testar a saída no Word uma vez que esse é o software que a maioria de seus visitantes utilizará.

Software para criar modelo PDF

Os documentos PDF são um pouco mais difíceis de gerar. A maneira mais fácil é adquirir o Adobe Acrobat. Esse software permitirá criar PDFs de alta qualidade a partir de várias aplicações. Utilizamos Acrobat para criar o arquivo de modelo para esse projeto.

Para criar o arquivo, utilizamos Microsoft Word a fim de projetar um documento. Uma das ferramentas no pacote Acrobat é o Adobe Distiller. Dentro do Distiller, precisamos selecionar algumas opções não-padrão. O arquivo deve ser armazenado no formato ASCII e a compactação precisa ser desativada. Depois que esses são configurados, criar um arquivo PDF é tão fácil quanto imprimir.

Você pode descobrir mais sobre o Acrobat em:

http://www.adobe.com/products/acrobat/

Voce pode comprá-lo on-line ou de um revendedor comum de software.

Outra opção para criar PDF é o programa de conversão `ps2pdf`, que, como o nome sugere, converte arquivos Postscript em arquivos PDF. Esse tem a vantagem de ser livre, mas nem sempre produz uma boa saída para documentos com imagens ou fontes não-padrão. O conversor `ps2pdf` vem com o pacote Ghostscript mencionado anteriormente.

Obviamente, se você vai criar um arquivo PDF dessa maneira, precisará criar um arquivo Postscript primeiro. Os usuários do Unix em geral utilizarão os utilitários `a2ps` ou `dvips` para esse propósito.

Se estiver trabalhando em um ambiente Windows, você também pode criar arquivos Postscript sem Adobe Distiller, embora via um processo ligeiramente mais complicado. Você precisará instalar um driver de impressora PostScript. Por exemplo, você pode utilizar o driver Apple LaserWriter IINT. Se não tiver um driver PostScript instalado, você pode fazer download de um no site da Adobe em:

http://www.adobe.com/support/downloads/product.jsp?product=44&platform=Windows

Para criar seu arquivo PostScript, você precisará selecionar essa impressora e a opção Print to File, em geral localizada na caixa de diálogo Print.

A maioria das aplicações Windows produzirá um arquivo com uma extensão `.prn`, que deve ser um arquivo PostScript. Você provavelmente deve renomeá-lo para ser um arquivo `.ps`. Visualize-o utilizando o GSview ou outro visualizador PostScript ou crie um arquivo PDF com o utilitário `ps2pdf`.

Esteja ciente de que diferentes drivers de impressora produzem saída de PostScript de qualidade variada. Talvez você ache que alguns arquivos PostScript que você produz fornecem erros quando executam pelo utilitário `ps2pdf`. Sugerimos utilizar um driver de impressora diferente.

Se você pretende criar apenas um pequeno número de arquivos PDF, o serviço on-line da Adobe talvez atenda bem às suas necessidades. Por US$9,90 mensais, você pode carregar arquivos em vários formatos e fazer download de um arquivo PDF. O serviço funcionou bem para nosso certificado, mas não permite selecionar opções que são importantes para esse projeto. O PDF criado será armazenado como um arquivo binário e compactado. Isso o torna muito difícil de modificar.

Esse serviço pode ser localizado em:

http://createpdf.adobe.com/

Há uma opção de avaliação grátis para esse serviço se você quiser testá-lo.

Há também uma interface baseada em FTP grátis para `ps2pdf` na Net Distillery:

http://www.babinszki.com/distiller/

Uma opção final seria codificar em XML e utilizar XSLT (XML Style Sheet Transformations) para convertê-lo a PDF e qualquer outro formato desejado. Esse método requer um bom entendimento de XSLT, e não é tratado aqui.

Software para criar PDF programaticamente

Suporte para criar documentos PDF está disponível a partir de dentro do PHP. Duas bibliotecas de funções diferentes estão disponíveis, com intenções semelhantes. Como elas contam com bibliotecas externas, nenhuma é compilada para o PHP por padrão.

As funções PDFlib do PHP, utilizam a biblioteca PDFlib, disponível em:
http://www.pdflib.com

As funções ClibPDF utilizam a biblioteca ClibPDF, disponível em:
http://www.fastio.com/

Essas duas bibliotecas são semelhantes. Elas fornecem uma API de funções para gerar um documento PDF. Escolhemos utilizar PDFlib porque esta parece ser atualizada e conservada mais regularmente.

Vale notar que nenhuma dessas bibliotecas é Free Software. Ambas permitem alguma utilização não-comercial sem cobrança, mas exigem uma taxa de licença se você pretende fornecer um serviço comercial que as utilize.

Algumas bibliotecas gratuitas, como FPDF, estão começando a se tornar disponíveis. FPDF ainda não é tão rica em recursos quanto as bibliotecas comerciais. Além disso, como FPDF é escrita em PHP (e não em C como extensão do PHP), é mais lenta do que as outras duas. Você pode fazer o download da FPDF em:
http://www.fpdf.org/

Neste capítulo, utilizamos a PDFlib porque provavelmente é a extensão mais utilizada.

Você pode ver se a PDFlib já está instalada em seu sistema verificando a saída da função `phpinfo( )`. Sob o título `pdf`, você pode descobrir se o suporte de PDFlib está ativado, bem como a versão utilizada.

Se pretende utilizar imagens TIFF ou JPEG nos documentos PDF, você também precisará instalar a biblioteca TIFF, disponível em:
http://www.libtiff.org/

e a biblioteca JPEG, disponível em:
ftp://ftp.uu.net/graphics/jpeg/

Em sistemas Unix, a PDFlib agora também contém bibliotecas de objetos compartilhados pré-construídas para PHP 4.3 e 5.0. Elas podem ser carregadas de `php.ini` ou com `dl( )`; portanto, não há necessidade de recompilar para acrescentar essa funcionalidade.

Em um servidor Windows, o PDFlib DLL é empacotado no arquivo PHP Zip, assim você simplesmente precisará remover o caractere de comentário da extensão no seu arquivo `php.ini`.

Visão geral da solução

Produziremos um sistema com três possíveis resultados. Como mostrado na Figura 32.1, faremos perguntas de questionário, avaliaremos as respostas e então geraremos um certificado de uma destas três maneiras:

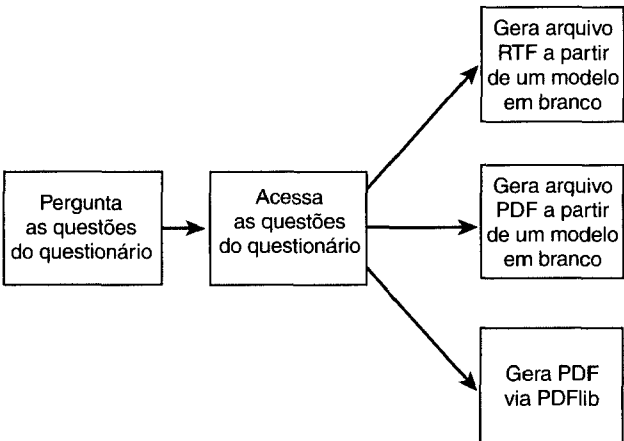



Figura 32.1 Nosso sistema de certificação gerará um dos três certificados diferentes.

- Geraremos um documento RTF a partir de um modelo em branco.
- Geraremos um documento PDF a partir de um modelo em branco.
- Geraremos um documento PDF programaticamente via PDFlib.

Um resumo dos arquivos no projeto de certificação é mostrado na Tabela 32.1.

Tabela 32.1 **Arquivos na aplicação de certificação**

Nome	Tipo	Descrição
index.html	Página HTML	Formulário HTML que contém as perguntas do questionário.
score.php	Aplicação	Script para avaliar as respostas dos usuários.
rtf.php	Aplicação	Script para gerar certificado RTF a partir do modelo.
pdf.php	Aplicação	Script para gerar certificado PDF a partir do modelo.
pdflib.php	Aplicação	Script para gerar certificado PDF utilizando PDFlib.
signature.png	Imagem	Imagem de mapa de bits da assinatura para ser incluída no certificado de PDFlib.
PHPCertification.rtf	RTF	Modelo de certificado RTF.
PHPCertification.pdf	PDF	Modelo de certificado PDF.

Vamos seguir em frente e examinar a aplicação.

Fazendo as perguntas

O arquivo index.html é simples e direto. Esse arquivo precisa conter um formulário em HTML que pergunte ao usuário seu nome e a resposta para várias perguntas. Em uma aplicação de avaliação real, provavelmente recuperaríamos essas perguntas a partir de um banco de dados. Aqui estamos focalizando a produção do certificado, então apenas codificaremos manualmente algumas perguntas no próprio código de HTML.

O campo name é uma entrada de texto. Cada pergunta tem três botões de opção para permitir ao usuário indicar sua resposta preferida. O formulário tem um botão de imagem como um botão submit.

O código para essa página é mostrado na Listagem 32.1.

Listagem 32.1 **index.html – Página HTML contendo perguntas do questionário**

```
<html>
  <body>
    <h1><p align="center">
          <img src="rosette.gif" alt="">
         Certification
          <img src="rosette.gif" alt=""></p></h1>
    <p>You too can earn your highly respected PHP certification
       from the world famous Fictional Institute of PHP Certification.</p>
    <p>Simply answer the questions below:</p>

    <form action="score.php" method="post">

      <p>Your Name <input type="text" name="name"></p>
```

Listagem 32.1 Continuação

```
      <p>What does the PHP statement echo do?</p>
      <ol>
        <li><input type="radio" name="q1" value="1">
            Outputs strings.</li>
        <li><input type="radio" name="q1" value="2">
            Adds two numbers together.</li>
        <li><input type="radio" name="q1" value="3">
            Creates a magical elf to finish writing your code.</li>
      </ol>

      <p>What does the PHP function cos( ) do?</p>
      <ol>
        <li><input type="radio" name="q2" value="1">
            Calculates a cosine in radians.</li>
        <li><input type="radio" name="q2" value="2">
            Calculates a tangent in radians.</li>
        <li><input type="radio" name="q2" value="3">
            It is not a PHP function it is a lettuce.</li>
      </ol>

      <p>What does the PHP function mail( ) do?</p>
      <ol>
        <li><input type="radio" name="q3" value="1">
            Sends a mail message.
        <li><input type="radio" name="q3" value="2">
            Checks for new mail.
        <li><input type="radio" name="q3" value="3">
            Toggles PHP between male and female mode.
      </ol>

      <p align="center"><input type="image" src="certify-me.gif" border="0"></p>

    </form>
  </body>
</html>
```

O resultado do carregamento `index.html` em um navegador Web é mostrado na Figura 32.2.

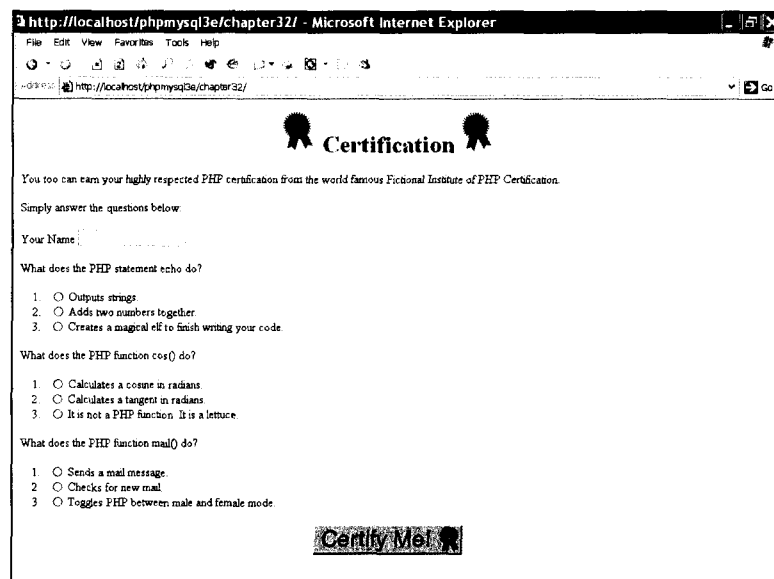



Figura 32.2 A página `index.html` pede ao usuário para responder às perguntas do questionário.

Graduando as respostas

Quando o usuário enviar suas respostas às perguntas no index.html, precisamos avaliá-lo e calcular uma pontuação. Isso é feito pelo script denominado score.php. O código para esse script é mostrado na Listagem 32.2.

Listagem 32.2 **score.php** – **Script para dar notas aos exames**

```
<?php
 //cria nomes de variáveis abreviados
 $q1 = $_POST['q1'];
 $q2 = $_POST['q2'];
 $q3 = $_POST['q3'];
 $name = $_POST['name'];

 // verifica se todos os dados foram recebidos
 if($q1==''||$q2==''||$q3==''||$name=='')
 {
   echo '<h1><p align = "center"><img src="rosette.gif" alt="">
                             Sorry:
                             <img src="rosette.gif" alt=""></p></h1>';
   echo '<p>You need to fill in your name and answer all questions</p>';
 }
 else
 {
   //acrescenta os escores
   $score = 0;
   if($q1 == 1) // a resposta correta de q1 é 1
     $score++;
   if($q2 == 1) // a resposta correta de q2 é 1
     $score++;
   if($q3 == 1) // a resposta correta de q3 é 1
     $score++;

   //converte pontuação para um porcentagem
   $score = $score / 3 * 100;

   if($score < 50)
   {
     // essa pessoa não passou
     echo '<h1 align="center"><img src="rosette.gif" alt="" />
                             Sorry:
                             <img src="rosette.gif" alt="" /></h1>';
     echo '<p>You need to score at least 50% to pass the exam</p>';
   }
   else
   {
     // cria uma string contendo a pontuação até uma casa decimal
     $score = number_format($score, 1);

     echo '<h1 align="center"><img src="rosette.gif" alt="" />
                             Congratulations
                             <img src="rosette.gif" alt="" /></h1>';
     echo "<p>Well done $name, with a score of $score%,
             you have passed the exam.</p>";

     // fornece links a scripts que geram os certificados
     echo '<p>Please click here to download your certificate as
             a Microsoft Word (RTF) file.</p>';
     echo '<form action="rtf.php" method="post">';
     echo '<center>
```

Listagem 32.2 **Continuação**

```
            <input type="image" src="certificate.gif" border="0">
          </center>';
    echo '<input type="hidden" name="score" value="'.$score.'">';
    echo '<input type="hidden" name="name" value="'.$name.'">';
    echo '</form>';

    echo '<p>Please click here to download your certificate as
             a Portable Document Format (PDF) file.</p>';
    echo '<form action="pdf.php" method="post">';
    echo '<center>
            <input type="image" src="certificate.gif" border="0">
          </center>';
    echo '<input type="hidden" name="score" value="'.$score.'">';
    echo '<input type="hidden" name="name" value="'.$name.'">';
    echo '</form>';

    echo '<p>Please click here to download your certificate as
             a Portable Document Format (PDF) file generated with PDFLib.</p>';
    echo '<form action="pdflib.php" method="post">';
    echo '<center>
            <input type="image" src="certificate.gif" border="0">
          </center>';
    echo '<input type=<@0148>hidden" name="score" value="'.$score.'">';
    echo '<input type="hidden" name="name" value="'.$name.'">';
    echo '</form>';
  }
}
?>
```

Esse script exibirá uma mensagem se o usuário não responder a todas as perguntas ou fizer uma pontuação menor que a nossa nota de aprovação escolhida.

Se o usuário respondeu às perguntas com sucesso, ele terá permissão para gerar um certificado. A saída de uma visita bem-sucedida é mostrada na Figura 32.3.

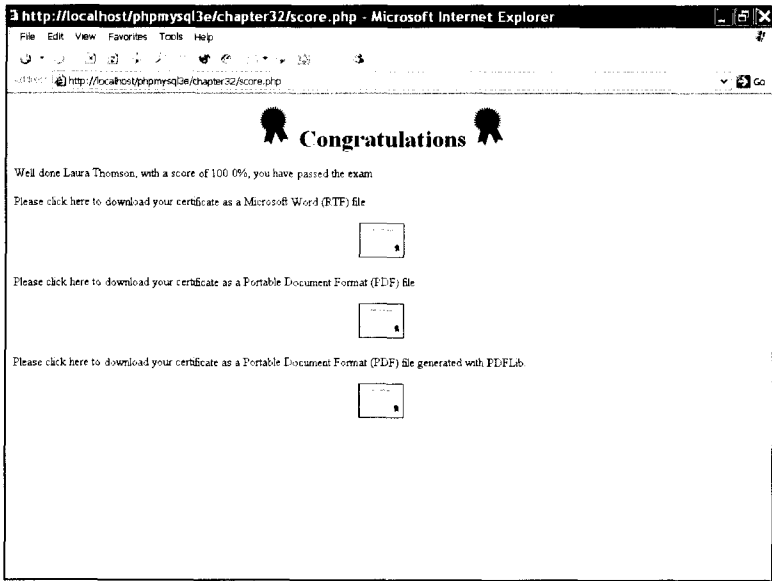



Figura 32.3 O script score.php apresenta aos visitantes bem-sucedidos a opção de gerar um certificado em uma de três maneiras.

Daqui para a frente, o usuário tem três opções. Ele pode ter um certificado RTF ou um dos dois certificados PDF. Veremos o script responsável por cada um.

Gerando um certificado RTF

Não há nada que nos impeça de gerar um documento RTF escrevendo texto ASCII para um arquivo ou uma variável alfanumérica, mas isso significaria aprender ainda outro conjunto de sintaxe.

Eis um documento RTF muito simples:

```
{\rtf1
{\fonttbl {\f0 Arial;}{\f1 Times New Roman;}}
\f0\fs28 Heading\par
\f1\fs20 This is an rtf document.\par
}
```

Esse documento configura uma tabela de fontes com duas fontes: Arial, referida como f0, e Times New Roman, referida como f1. Esse documento então escreve o Título (Heading) utilizando f0 (Arial) no tamanho de 28 (14 pontos). O controle \par indica uma quebra de parágrafo. Em seguida, escrevemos This is an rtf document usando f1 (Times New Roman) no tamanho 20 (10 pontos).

Poderíamos gerar um documento como esse manualmente, mas não há funções de economia de trabalho predefinidas para o PHP para tornar as partes mais difíceis, como incorporar imagens gráficas, mais fáceis. Felizmente, em muitos documentos, a estrutura, o estilo e a maior parte do texto são estáticos e somente as pequenas partes mudam de pessoa para pessoa. Uma maneira mais eficiente de gerar um documento é utilizando um modelo.

Podemos construir um documento complexo, como o mostrado na Figura 32.4, facilmente utilizando um processador de texto.

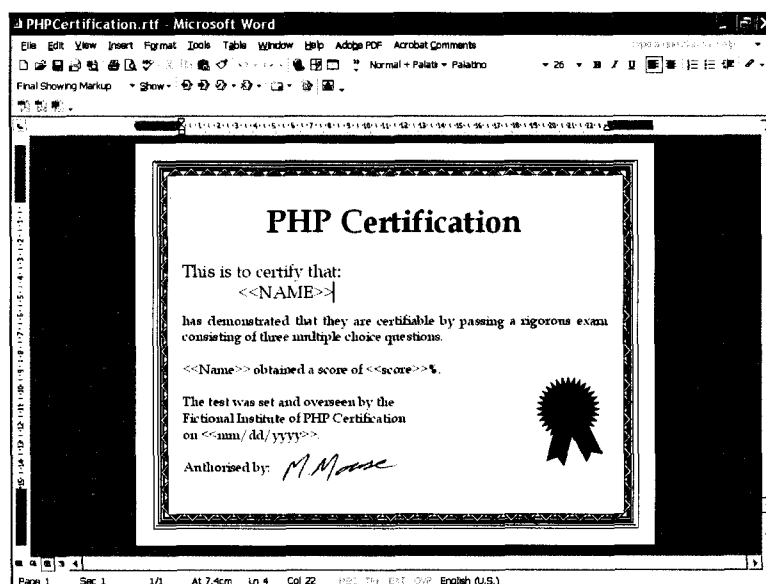



Figura 32.4 Utilizando um processador de texto, podemos criar facilmente um modelo complexo e atraente.

Nosso modelo inclui marcadores de lugar como <<NAME>> para marcar os lugares nos quais dados dinâmicos serão inseridos. Não é importante como são esses marcadores de lugar. Estamos utilizando uma descrição significativa entre dois conjuntos de sinais de maior e menor. É muito importante escolher marcadores de lugar que não apareçam acidentalmente no restante do documento. Isso ajudará a planejar seu modelo se os marcadores de lugar tiverem aproximadamente o mesmo comprimento que os dados pelos quais eles serão substituídos.

Os marcadores de lugar nesse documento são <<NAME>>, <<Name>>, <<score>> e <<mm/dd/yyyy>>. Observe que estamos utilizando tanto NAME como Name, porque pretendemos utilizar um método com distinção de letras maiúsculas e minúsculas para substituí-los.

Agora que temos um modelo, precisamos de um script para personalizá-lo. Esse script é chamado rtf.php, e seu código é mostrado na Listagem 32.3.

Listagem 32.3 **rtf.php – Script para produzir um certificado personalizado de RTF**

```
<?php
 //cria nomes de variáveis abreviados
 $name = $_POST['name'];
 $score = $_POST['score'];
 // verifica se temos os parâmetros necessários
 if( !$name || !$score )
 {
   echo '<h1>Error:</h1><p>This page was called incorrectly</p>';
 }
 else
 {
   //gera os cabeçalhos para ajudar ao navegador escolher a aplicação correta
   header( 'Content-Type: application/msword' );
   header( 'Content-Disposition: inline, filename=cert.rtf');

   $date = date( 'F d, Y' );

   // abre nosso arquivo de modelo
   $filename = 'PHPCertification.rtf';
   $output = file_get_contents($filename);

   // substitui os marcadores de lugar no modelo por nossos dados
   $output = str_replace( '<<NAME>>', strtoupper( $name ), $output );
   $output = str_replace( '<<Name>>', $name, $output );
   $output = str_replace( '<<score>>', $score, $output );
   $output = str_replace( '<<mm/dd/yyyy>>', $date, $output );

   // envia o documento gerado ao navegador
   echo $output;
 }
?>
```

Esse script realiza alguma verificação básica de erros para certificar-se de que todos os detalhes do usuário foram passados e então se move para a parte da criação do certificado.

A saída desse script será um arquivo RTF em vez de um arquivo HTML, então precisamos alertar o navegador do usuário sobre esse fato. Isso é importante para que o navegador possa tentar abrir o arquivo com a aplicação correta ou dar uma caixa de diálogo do tipo Save As... se ele não reconhecer a extensão .rtf.

Especificamos o tipo MIME do arquivo que estamos enviando para saída utilizando a função header( ) do PHP para enviar o cabeçalho de HTTP apropriado da seguinte maneira:

```
header('Content-type: application/msword');
header('Content-Disposition: inline, filename=cert.rtf');
```

O primeiro cabeçalho diz ao navegador que estamos enviando um arquivo do Microsoft Word (não estritamente verdadeiro, mas a aplicação auxiliar mais provável para abrir o arquivo RTF).

O segundo cabeçalho diz ao navegador para automaticamente exibir o conteúdo do arquivo e que seu nome de arquivo sugerido é cert.rtf. Esse é o nome de arquivo padrão que o usuário verá se tentar salvar o arquivo a partir de dentro do seu navegador.

Depois que os cabeçalhos forem enviados, abrimos e lemos nosso arquivo RTF de modelo para a variável `$output` e utilizamos a função `str_replace( )` para substituir nossos marcadores de lugar pelos dados reais que desejamos que apareçam no arquivo. A linha

```
$output = str_replace( '<<Name>>', $name, $output );
```

substituirá quaisquer ocorrências de `<<Name>>` do marcador de lugar pelo conteúdo da variável `$name`.

Tendo feito nossas substituições, é só uma questão de ecoar a saída para o navegador. Um resultado de exemplo desse script é mostrado na Figura 32.5.

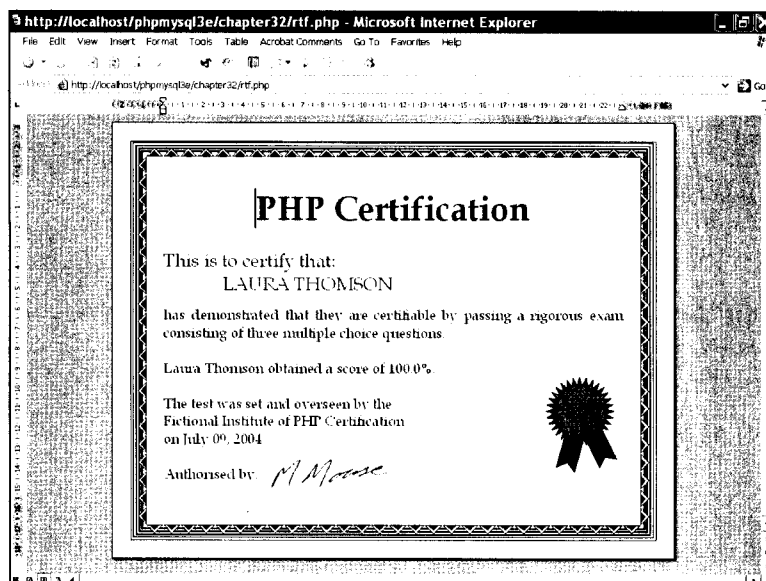



Figura 32.5 O script `rtf.php` gera um certificado a partir de um modelo RTF.

Essa abordagem funciona muito bem. As chamadas a `str_replace( )` executam muito rapidamente, mesmo que nosso modelo e, portanto, o conteúdo de `$output` sejam relativamente longos. O problema principal do ponto de vista dessa aplicação é que o usuário carregará o certificado em seu processador de texto para impressão. Isso provavelmente é um convite para as pessoas modificarem a saída. O RTF não permite criar um documento apenas de leitura.

Gerando um certificado PDF a partir de um modelo

O processo de gerar um certificado PDF a partir de um modelo é muito semelhante. A diferença principal é que quando criamos o arquivo PDF, alguns de nossos marcadores de lugar talvez sejam intercalados com códigos de formatação. Por exemplo, se olharmos o arquivo de modelo de certificado que criamos, podemos ver que os marcadores de lugar agora são semelhantes a:

```
<<N)-13(AME)-10(>)-6(>
<<Na)-9(m)0(e)-18(>>
<)-11(<)1(sc)-17(or)-6(e)-6(>)-11(>
<)-11(<)1(m)-12(m)0(/d)-6(d)-19(/)1(yy)-13(yy)-13(>>
```

Se olhar atentamente o arquivo, você verá que, diferente do RTF, esse não é um formato que humanos podem ler facilmente.

Nota O formato do modelo PDF pode variar dependendo da versão de Acrobat ou da geração de outra ferramenta PDF que você esteja usando. O código fornecido nesse exemplo pode não funcionar como esperado quando você gerar seus próprios modelos. Verifique seu modelo e altere o código para se adequar. Se ainda tiver problemas, use o exemplo PDFlib mostrado anteriormente neste capítulo.

Há algumas maneiras diferentes de lidar com isso. Poderíamos passar por cada um desses marcadores de lugar e excluir o código de formatação. Isso realmente faz pouca diferença sobre como é o documento no final uma vez que os códigos embutidos no modelo anterior indicam quanto espaço deve ser deixado entre as letras dos marcadores de lugar que vamos substituir de qualquer modo. Entretanto, se adotarmos essa abordagem, devemos avançar e auxiliar a edição do arquivo PDF e repetir isso toda vez que alterarmos ou atualizarmos o arquivo. Isso não é grande questão ao lidar apenas com quatro marcadores de lugar, mas torna-se um pesadelo quando, por exemplo, você tiver vários documentos com muitos marcadores de lugar e decidir alterar o cabeçalho em todos os documentos.

Podemos evitar esse problema utilizando uma técnica diferente. Você pode utilizar Adobe Acrobat para criar um formulário PDF – semelhante a um formulário HTML com campos de nomes em branco. Você então pode utilizar um script de PHP para criar o que é chamado um arquivo FDF (Forms Data Format), que é basicamente um conjunto de dados a ser mesclado com um modelo. Você pode criar FDFs utilizando a biblioteca de funções de FDF do PHP: especificamente, a função `fdf_create( )` para criar um arquivo, a função `fdf_set_value( )` para configurar os valores de campo, e a função `fdf_set_file( )` para configurar o arquivo associado do formulário modelo. Você então pode passar esse arquivo de volta para o navegador com o tipo MIME apropriado, nesse caso, `vnd.fdf` – e o plug-in Acrobat Reader do navegador devem substituir os dados no formulário.

Essa é uma maneira elegante, mas tem duas limitações. Primeiro, ela assume que você possui uma cópia do Acrobat. Segundo, é difícil substituir em texto que é in-line em vez de texto que se parece com um campo de formulário. Isso pode ou não representar um problema, dependendo do que você está tentando fazer. Utilizamos amplamente a geração PDF para gerar cartas nas quais muitos itens devem ser substituídos in-line. Os FDFs não funcionam bem para esse propósito. Se, por exemplo, estiver autopreenchendo um formulário de imposto on-line, isso não será um problema.

Você pode ler mais sobre o formato FDF no site da Adobe:

http://partners.adobe.com/asn/developer/acrosdk/forms.html

Você também deve ver a documentação de FDF no manual de PHP se decidir utilizar essa abordagem:

http://www.php.net/manual/en/ref.fdf.php

Voltemos agora à nossa solução de PDF para o problema anterior.

Ainda podemos localizar e substituir os marcadores de lugar em nosso arquivo PDF se reconhecermos que os códigos adicionais de formatação consistem unicamente em hifens, dígitos e parênteses e podem, portanto, ser correspondidos via uma expressão regular. Escrevemos uma função, `pdf_replace( )`, para automaticamente gerar uma expressão regular correspondente para um marcador de lugar e substituir esse marcador de lugar pelo texto apropriado.

Perceba que em algumas versões do Acrobat, os marcadores de lugar estão em texto simples e podem ser substituídos por `str_replace`, como fizemos anteriormente.

Além dessa adição, o código para gerar o certificado via um modelo PDF é muito semelhante à versão RTF. Esse script é mostrado na Listagem 32.4.

Listagem 32.4 **`pdf.php` – Script para produzir certificado PDF personalizado via um modelo**

```
<?php
set_time_limit( 180 ); // este script pode ser lento

//cria nomes de variável abreviados
$name = $_POST['name'];
$score = $_POST['score'];

function pdf_replace( $pattern, $replacement, $string )
{
  $len = strlen( $pattern ) ;
  $regex = '';
```

Listagem 32.4 **Continuação**

```
    for ( $i = 0; $i<$len; $i++ )
    {
      $regex .= $pattern[$i];
      if ($i<$len-1)
        $regex .= '(\)-?[0-9]+\( )?';
    }
    return ereg_replace ( $regex, $replacement, $string );
  }

  if(!$name||!$score)
  {
    echo '<h1>Error:</h1><p>This page was called incorrectly</p>';
  }
  else
  {
    //gera os cabeçalhos para ajudar o navegador a escolher a aplicação certa
    header( 'Content-Disposition:  filename=cert.pdf');
    header( 'Content-Type: application/pdf' );

    $date = date( 'F d, Y' );

    // abre nosso arquivo modelo
    $filename = 'PHPCertification.pdf';
    $output = file_get_contents($filename);

    // substitui os marcadores de lugar no modelo pelos nossos dados
    $output = pdf_replace( '<<NAME>>', strtoupper( $name ), $output );
    $output = pdf_replace( '<<Name>>', $name, $output );
    $output = pdf_replace( '<<score>>', $score, $output );
    $output = pdf_replace( '<<mm/dd/yyyy>>', $date, $output );

    // envia o documento gerado para o navegador

    echo $output;
  }
?>
```

Esse script produz uma versão personalizada de nosso documento PDF. O documento, mostrado na Figura 32.6, será impresso confiavelmente em numerosos sistemas e esse documento é mais difícil de o destinatário modificar ou editar. Você pode ver que o documento PDF na Figura 32.6 é quase idêntico ao documento RTF da Figura 32.5.

Um problema com essa abordagem é que o código executa muito lentamente por causa da expressão regular de correspondência exigida. Expressões regulares executam muito mais lentamente que `str_replace( )` que poderíamos utilizar para a versão RTF.

Se for corresponder a um número grande de marcadores de lugar ou tentar gerar muitos desses documentos no mesmo servidor, talvez você queira ver outras abordagens. Isso não seria um problema para um modelo mais simples. Grande parte do volume nesse arquivo são dados que representam as imagens.

Gerando um documento PDF utilizando PDFlib

A PDFlib é projetada para gerar documentos PDF dinâmicos via Web. Ela não é estritamente parte do PHP, mas antes uma biblioteca separada, com várias funções projetadas para serem chamadas a partir de uma ampla variedade de linguagens de programação. As vinculações de linguagem estão disponíveis para C, C++, Java, Perl, Python, Tcl e ActiveX/COM.

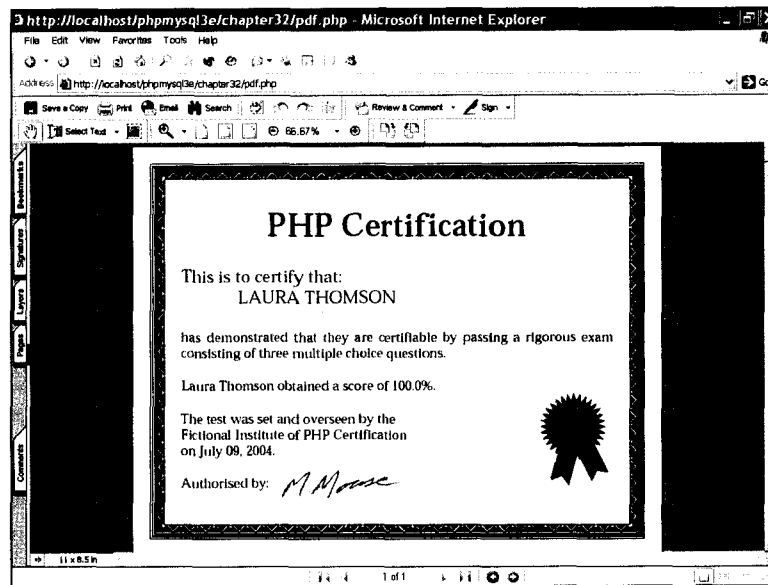



Figura 32.6 O script `pdf.php` gera um certificado a partir de um modelo PDF.

A partir do PHP 4.0.5, a PDFLib foi oficialmente suportada pela PDFlib GmbH. Isso significa que você pode referenciar a documentação do PHP em:

http://www.php.net/manual/ref.pdf.php

ou fazer o download da documentação oficial em pdflib.com.

Um script Hello World para PDFlib

Depois que tiver o PHP instalado com o PDFlib ativado, você pode testá-lo com um programa simples como o exemplo Hello World na Listagem 32.5.

Listagem 32.5 **`testpdf.php`** – **Exemplo clássico do Hello World utilizando PDFlib via PHP**

```
<?php

 // cria um documento pdf na memória
 $pdf = pdf_new( );
 pdf_open_file($pdf, '');

 pdf_set_info($pdf, "Creator", "pdftest.php");
 pdf_set_info($pdf, "Author", "Luke Welling and Laura Thomson");
 pdf_set_info($pdf, "Title", "Hello World (PHP)");

 // o formato US letter tem 11" x 8.5" e há aproximadamente 72 pontos por polegada
 pdf_begin_page($pdf, 8.5*72, 11*72);

 // adiciona um bookmark
 pdf_add_bookmark($pdf, 'Page 1', 0, 0);

 $font = pdf_findfont($pdf, 'Times-Roman', 'host', 0);
 pdf_setfont($pdf, $font, 24);
 pdf_set_text_pos($pdf, 50, 700);

 // escreve o texto
 pdf_show($pdf,'Hello,world!');
 pdf_continue_text($pdf,'(says PHP)');

 // finaliza o documento
```

Listagem 32.5 Continuação

```
 pdf_end_page($pdf);
 pdf_close($pdf);

 $data = pdf_get_buffer($pdf);

 // gera os cabeçalhos para ajudar ao navegador escolher a aplicação correta
 header('Content-Type: application/pdf');
 header('Content-Disposition: inline; filename=test.pdf');
 header('Content-Length: ' . strlen($data));

 // saída em PDF
 echo $data;

?>
```

O erro mais provável que você verá se esse script falhar é:

```
Fatal error: Call to undefined function pdf_new( )
in c:\program files\apache group\Apache\htdocs\phpmysql3e\chapter32\testpdf.php
on line 4
```

Isso significa que você não tem a extensão de PDFlib compilada no PHP.

A instalação é relativamente simples e direta, mas alguns detalhes mudam dependendo das versões exatas do PHP e PDFlib que você está utilizando. Um bom lugar para verificar sugestões detalhadas são as notas de colaboração do usuário na página PDFlib no manual anotado do PHP.

Quando tiver esse script pronto e funcionando em seu sistema, é hora de ver como funciona. As linhas

```
$pdf = pdf_new( );
pdf_open_file($pdf, '');
```

inicializam um documento PDF na memória.

A função `pdf_set_info( )` permite fazer uma tag para o documento com um campo de assunto, título, criador, autor, uma lista de palavras-chave e um campo definido pelo usuário personalizado.

Aqui estamos configurando um criador, autor e título. Observe que todos os seis campos de informações são opcionais.

```
pdf_set_info($pdf, 'Creator', 'pdftest.php');
pdf_set_info($pdf, 'Author', 'Luke Welling and Laura Thomson');
pdf_set_info($pdf, 'Title', 'Hello World (PHP)');
```

Um documento PDF consiste em várias páginas. Para iniciar uma nova página, precisamos chamar `pdf_begin_page`. Da mesma maneira como o identificador retornado por `pdf_open( )`, o `pdf_begin_page( )` requer as dimensões da página. Todas as páginas em um documento podem ter um tamanho diferente, mas a menos que tenha uma boa razão, você deve utilizar um tamanho de papel comum.

A PDFlib trabalha com pontos, tanto para tamanho de página como para posicionar as coordenadas em cada página. Para referência, o formato A4 tem aproximadamente 595 por 842 pontos e o formato de papel carta norte-americana (U.S. letter paper) tem 612 por 792 pontos. Isso significa que nossa linha

```
pdf_begin_page($pdf, 8.5*72, 11*72);
```

cria uma página em nosso documento, dimensionada para o formato de papel U.S. letter.

Um documento PDF não precisa ser simplesmente um documento imprimível. Muitos recursos PDF podem ser incluídos no documento como hyperlinks e bookmarks. A função `pdf_add_outline( )` adicionará um bookmark à estrutura de tópicos de documento. Os bookmarks em um docu-

mento aparecerão em um painel separado no Acrobat Reader, permitindo pular diretamente para seções importantes.

Esta linha

```
pdf_add_bookmark($pdf, 'Page 1', 0, 0);
```

adiciona uma entrada à estrutura de tópicos rotulada Page 1, que referenciará a página atual.

As fontes disponíveis nos sistemas variam de sistema operacional para sistema operacional e até de máquina individual para máquina. A fim de garantir resultados consistentes, um conjunto de fontes básicas funcionará com todo leitor PDF. As 14 fontes básicas são:

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats

As fontes fora desse conjunto podem ser embutidas em documentos, mas isso aumentará o tamanho do arquivo e talvez não seja aceitável para qualquer que seja a licença sob a qual você possui essa fonte particular. Podemos escolher uma fonte, seu tamanho e codificação de caractere da seguinte maneira:

```
$font = pdf_findfont($pdf, 'Times-Roman', 'host', 0);
pdf_setfont($pdf, $font, 24);
```

Os tamanhos da fonte são especificados em pontos. Escolhemos adotar a codificação de caractere. Os valores admissíveis são `winansi`, `builtin`, `macroman`, `ebcdic` ou `host`. Os significados dos valores diferentes são:

- `winansi` – ISO 8859-1 mais caracteres especiais adicionados pela Microsoft como o símbolo do Euro.
- `macroman` – Codificação Mac Roman. O conjunto de caracteres padrão do Macintosh.
- `ebcdic` – EBCDIC como utilizado em sistemas IBM AS/400.
- `builtin` – Utilize a codificação predefinida para a fonte. Normalmente utilizado com símbolos e fontes não-latinas.
- `host` – Seleciona automaticamente `macroman` em um Mac, `ebcdic` em um sistema baseado em EBCDIC e `winansi` em todos os outros sistemas.

Se você não precisar incluir caracteres especiais, a escolha de codificação não é importante.

Um documento PDF não é como um documento HTML ou um documento de processador de texto. O texto por padrão não inicia na parte superior esquerda e nem flui sobre as outras linhas como requerido. Precisamos escolher onde colocar cada linha de texto. Como já mencionado, o PDF utiliza pontos para especificar localizações. A origem (coordenada x, y [0, 0]) está no canto inferior esquerdo da página.

Considerando que nossa página tem 612 por 792 pontos, o ponto (50, 700) está aproximadamente a 1,6cm da esquerda da página e 2,5cm e 0,8cm da parte superior. Para configurar a posição do texto nesse ponto, utilizamos:

```
pdf_set_text_pos($pdf, 50, 700);
```

Por fim, tendo configurada a página, podemos escrever algum texto. Para adicionar texto na posição atual utilizando a fonte atual, utilizamos `pdf_show( )`.

A linha

```
pdf_show($pdf,'Hello,world!');
```

adiciona o texto `"Hello World!"` ao nosso documento.

Para mudar para a próxima linha e escrever mais texto, utilizamos `pdf_continue_text( )`. Para adicionar a string `"(says PHP)"`, utilizamos:

```
pdf_continue_text($pdf,'(says PHP)');
```

A localização exata em que isso aparecerá dependerá da fonte e do tamanho selecionados.

Se em vez de linhas ou frases você estiver utilizando parágrafos contíguos, talvez descubra que a função `pdf_show_boxed( )` é mais útil. Ela permite declarar e colocar texto em uma caixa de texto.

Quando terminamos de adicionar elementos a uma página, precisamos chamar `pdf_end_page( )` da seguinte maneira:

```
pdf_end_page($pdf);
```

Quando terminarmos o documento PDF inteiro, precisamos fechá-lo utilizando `pdf_close( )`. Quando estamos gerando um arquivo, também precisamos fechar o arquivo.

A linha

```
pdf_close($pdf);
```

completa a geração de documento Hello World.

Agora você pode enviar o PDF completo ao navegador:

```
$data = pdf_get_buffer($pdf);

// gera os cabeçalhos para ajudar ao navegador escolher a aplicação correta
header('Content-Type: application/pdf');
header('Content-Disposition: inline; filename=test.pdf');
header('Content-Length: ' . strlen($data));

// saída em PDF
echo $data;
```

Você também pode gravar esses dados para o disco se preferir. PDFlib permite fazer isso passando um nome de arquivo como segundo parâmetro a `pdf_open_file( )`. No momento em que escrevíamos este livro, esse recurso dava problemas no Windows (PHP 5.0.0). Se precisar gravar os dados para o disco, pode fazê-lo manualmente.

Esse exemplo foi tirado do exemplo na linguagem C contido na documentação PDFlib e deve ser um ponto de partida.

Observe que alguns parâmetros de função PDFlib documentados no manual do PHP como sendo opcionais são exigidos em algumas versões do PDFlib. O documento do certificado é mais com-

plicado, incluindo borda, imagem vetorial e imagem bitmap. Com as outras duas técnicas, você pode acrescentar esses recursos utilizando um processador de texto. Com a PDFlib, é preciso adicioná-los manualmente.

Gerando um certificado com PDFlib

A fim de utilizar o PDFlib, escolhemos fazer alguns compromissos. Embora seja quase certamente possível duplicar com exatidão o certificado que utilizamos antes, seria exigido bastante esforço para gerar e posicionar cada elemento manualmente em vez de utilizar uma ferramenta como Microsoft Word para ajudar a planejar o documento.

Estamos utilizando o mesmo texto que antes, incluindo a roseta vermelha e a assinatura de mapa de bits, mas não vamos duplicar a borda complexa. O código completo para esse script é mostrado na Listagem 32.6.

Listagem 32.6 **pdflib.php** – Gerando nosso certificado utilizando PDFlib

```
<?php

 // cria nomes abreviados de variáveis
 $name = $_POST['name'];
 $score = $_POST['score'];

 if(!$name||!$score)
 {
   echo '<h1>Error:</h1><p>This page was called incorrectly</p>';
   exit;
 }
 else
 {
   $date = date( 'F d, Y' );

   // cria um documento pdf na memória
   $pdf = pdf_new( );
   pdf_open_file($pdf, '');

   // configura nome da fonte para uso posterior
   $fontname = 'Times-Roman';

   // configura o tamanho da página em pontos e cria a página
   // o formato US letter tem 11" x 8.5" e existem aproximadamente 72 pontos por polegada
   $width = 11*72;
   $height = 8.5*72;
   pdf_begin_page($pdf, $width, $height);

   // desenha as bordas
   $inset = 20; // espaço entre a borda e a margem da página
   $border = 10; // largura da linha da borda principal
   $inner = 2; // espaço entre a borda

   //desenha borda externa
   pdf_rect($pdf, $inset-$inner,
                  $inset-$inner,
                  $width-2*($inset-$inner),
                  $height-2*($inset-$inner));
   pdf_stroke($pdf);

   //desenha borda principal com $border pontos de largura
   pdf_setlinewidth($pdf, $border);
   pdf_rect($pdf, $inset+$border/2,
```

Listagem 32.6 **Continuação**

```
               $inset+$border/2,
               $width-2*($inset+$border/2),
               $height-2*($inset+$border/2));
pdf_stroke($pdf);
pdf_setlinewidth($pdf, 1.0);

// desenha borda interna
pdf_rect($pdf, $inset+$border+$inner,
               $inset+$border+$inner,
               $width-2*($inset+$border+$inner),
               $height-2*($inset+$border+$inner));
pdf_stroke($pdf);

// acrescenta cabeçalho
$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font)
  pdf_setfont($pdf, $font, 48);
$startx = ($width - pdf_stringwidth($pdf, 'PHP Certification',
          $font, 48))/2;
pdf_show_xy($pdf, 'PHP Certification', $startx, 490);

// acrescenta texto
$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font)
  pdf_setfont($pdf, $font, 26);
$startx = 70;
pdf_show_xy($pdf, 'This is to certify that:', $startx, 430);
pdf_show_xy($pdf, strtoupper($name), $startx+90, 391);

$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font)
  pdf_setfont($pdf, $font, 20);

pdf_show_xy($pdf, 'has demonstrated that they are certifiable '.
                  'by passing a rigorous exam', $startx, 340);
pdf_show_xy($pdf, 'consisting of three multiple choice questions.',
                  $startx, 310);

pdf_show_xy($pdf, "$name obtained a score of $score".'%.', $startx, 260);

pdf_show_xy($pdf, 'The test was set and overseen by the ', $startx, 210);
pdf_show_xy($pdf, 'Fictional Institute of PHP Certification',
                  $startx, 180);
pdf_show_xy($pdf, "on $date.", $startx, 150);
pdf_show_xy($pdf, 'Authorized by:', $startx, 100);

// acrescenta a imagem de assinatura do bitmap
// você pode precisar mudar o caminho para o arquivo de assinatura aqui
$path = 'C:/Program Files/Apache Group/Apache/htdocs/phpmysql3e/chapter32/';

// utilizar a versão gif como PDFLib para o Windows parece ter problemas com GIFs
$signature = pdf_open_image_file($pdf, 'gif', $path.'signature.gif',
                                 'mask', 0);
pdf_place_image($pdf, $signature, 200, 75, 1);
pdf_close_image($pdf, $signature);

// configura cores da roseta
pdf_setcolor($pdf, 'fill', 'rgb', 0, 0, .4, 0);  // azul-escuro
pdf_setcolor($pdf, 'stroke', 'rgb', 0, 0, 0, 0); // preto
```

Listagem 32.6 Continuação

```
    // desenha a fita 1
    pdf_moveto($pdf, 630, 150);
    pdf_lineto($pdf, 610, 55);
    pdf_lineto($pdf, 632, 69);
    pdf_lineto($pdf, 646, 49);
    pdf_lineto($pdf, 666, 150);
    pdf_closepath($pdf);
    pdf_fill($pdf);

    // desenha a fita 1
    pdf_moveto($pdf, 630, 150);
    pdf_lineto($pdf, 610, 55);
    pdf_lineto($pdf, 632, 69);
    pdf_lineto($pdf, 646, 49);
    pdf_lineto($pdf, 666, 150);
    pdf_closepath($pdf);
    pdf_stroke($pdf);

    // desenha a fita 2
    pdf_moveto($pdf, 660, 150);
    pdf_lineto($pdf, 680, 49);
    pdf_lineto($pdf, 695, 69);
    pdf_lineto($pdf, 716, 55);
    pdf_lineto($pdf, 696, 150);
    pdf_closepath($pdf);
    pdf_fill($pdf);

    // desenha a fita 2
    pdf_moveto($pdf, 660, 150);
    pdf_lineto($pdf, 680, 49);
    pdf_lineto($pdf, 695, 69);
    pdf_lineto($pdf, 716, 55);
    pdf_lineto($pdf, 696, 150);
    pdf_closepath($pdf);
    pdf_stroke($pdf);
    pdf_setcolor($pdf, 'fill', 'rgb', .8, 0, 0, 0); // red

    // desenha a roseta
    draw_star(665, 175, 32, 57, 10, $pdf, true);

    // desenha a roseta
    draw_star(665, 175, 32, 57, 10, $pdf, false);

    // termina a página e prepara para saída
    pdf_end_page($pdf);
    pdf_close($pdf);
    $data = pdf_get_buffer($pdf);

    // gera os cabeçalhos para ajudar ao navegador escolher a aplicação correta
    header('Content-Type: application/pdf');
    header('Content-Disposition: inline; filename=certificate.pdf');
    header('Content-Length: ' . strlen($data));

    // saída em PDF
    echo $data;
}

function draw_star($centerx, $centery, $points, $radius,
                   $point_size, $pdf, $filled)
{
```

Listagem 32.6 **Continuação**

```
  $inner_radius = $radius-$point_size;

  for ($i = 0; $i<=$points*2; $i++ )
  {
    $angle= ($i*2*pi( ))/($points*2);

    if($i%2)
    {
      $x = $radius*cos($angle) + $centerx;
      $y = $radius*sin($angle) + $centery;
    }
    else
    {
      $x = $inner_radius*cos($angle) + $centerx;
      $y = $inner_radius*sin($angle) + $centery;
    }
    if($i==0)
      pdf_moveto($pdf, $x, $y);
    else if($i==$points*2)
      pdf_closepath($pdf);
    else
      pdf_lineto($pdf, $x, $y);
  }
  if($filled)
    pdf_fill_stroke($pdf);
  else
    pdf_stroke($pdf);
}
?>
```

O certificado produzido utilizando esse script é mostrado na Figura 32.7. Como você pode ver, ele é bem semelhante aos outros, exceto que a borda é mais simples e a estrela é um pouco diferente. Isso porque as desenhamos no documento em vez de utilizar um arquivo de clip-art existente.

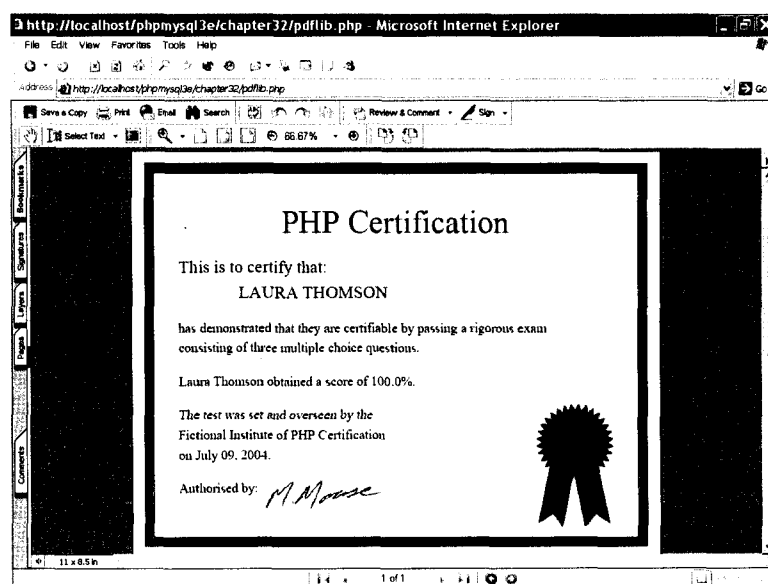



Figura 32.7 O script pdflib.php desenha o certificado em um documento PDF.

Veremos algumas partes desse script que são diferentes dos exemplos anteriores.

Os visitantes precisam colocar seus próprios detalhes em um certificado, então criaremos o documento na memória em vez de em um arquivo. Se gravássemos em um arquivo, precisaríamos nos preocupar com mecanismos para criar nomes de arquivo únicos, impedir que as pessoas espionassem os certificados de outras e determinar uma maneira de excluir arquivos de certificado mais antigos para liberar espaço de unidade de disco no servidor. A fim de criar um documento na memória, chamamos `pdf_new( )`, sem parâmetros, seguido por uma chamada a `pdf_open_file( )` como a seguir:

```
$pdf = pdf_new( );
pdf_open_file($pdf);
```

Nossa borda simplificada consistirá em três listras: uma borda espessa e duas bordas finas, uma dentro da borda principal e uma fora. Desenharemos todas essas como retângulos.

Para posicionar as bordas de maneira como possamos facilmente alterar o tamanho da página ou a aparência das bordas, basearemos todas as posições de borda nas variáveis que já temos, `$width` e `$height` e algumas novas: `$inset`, `$border` e `$inner`. Utilizaremos `$inset` para especificar quantos pontos de largura terá a borda na margem da página, `$border` para especificar a espessura da borda principal e `$inner` para especificar a largura da lacuna entre a borda principal e as bordas finas.

Se você desenhou com outras imagens gráficas API, desenhar com PDFlib apresentará poucas surpresas. Se não leu o Capítulo 21, talvez você ache útil fazer isso agora, uma vez que desenhar imagens com a biblioteca `gd` é bem semelhante a desenhar imagens com a PDFlib.

As bordas finas são fáceis. Para criar um retângulo, utilizamos `pdf_rect( )`, que requer como parâmetros o identificador de documento PDF, a coordenada x e y do canto esquerdo do retângulo e a largura e a altura do retângulo. Como queremos que nosso layout seja flexível, calculamos essas a partir das variáveis que configuramos.

```
pdf_rect($pdf, $inset-$inner,
               $inset-$inner,
               $width-2*($inset-$inner),
               $height-2*($inset-$inner));
```

A chamada a `pdf_rect( )` configura um caminho na forma de um retângulo. Para desenhar essa forma, precisamos chamar a função `pdf_stroke( )` da seguinte maneira:

```
pdf_stroke($pdf);
```

Para desenhar a borda principal, precisamos especificar a largura da linha. A largura da linha padrão é de 1 ponto. A seguinte chamada a `pdf_setlinewidth( )` irá configurá-la como `$border` pontos (nesse caso 10):

```
pdf_setlinewidth($pdf, $border);
```

Com a largura configurada, novamente criamos um retângulo com `pdf_rect( )` e chamamos `pdf_stroke( )` para desenhá-lo.

```
pdf_rect($pdf, $inset+$border/2,
               $inset+$border/2,
               $width-2*($inset+$border/2),
               $height-2*($inset+$border/2));
pdf_stroke($pdf);
```

Depois que desenhamos a largura da linha, precisamos lembrar de configurar a largura da linha outra vez como 1 com esse código:

```
pdf_setlinewidth($pdf, 1.0);
```

Vamos utilizar `pdf_show_xy( )` para posicionar cada linha de texto no certificado. Para a maioria das linhas de texto, utilizamos uma margem esquerda configurável (`$startx`) como a coordenada x e um valor escolhido aleatoriamente como a coordenada y. Como queremos o título centralizado

na página, precisamos saber sua largura a fim de posicioná-lo no lado esquerdo. Podemos obter a largura utilizando `pdf_stringwidth( )`. A chamada

```
pdf_stringwidth($pdf, 'PHP Certification', $font, 48);
```

retornará a largura da string "`PHP Certification`" na fonte e no tamanho da fonte atuais.

Como com as outras versões do certificado, incluiremos uma assinatura como um mapa de bits digitalizado. As instruções

```
$signature = pdf_open_image_file($pdf, 'gif', $path.'signature.gif',
                                  'mask', 0);
pdf_place_image($pdf, $signature, 200, 75, 1);
pdf_close_image($pdf, $signature);
```

abrem um arquivo GIF contendo a assinatura, acrescentam a imagem à página no local especificado e fecham o arquivo GIF. Outros tipos de arquivo também podem ser usados. (Voltamos a usar GIF aqui porque o suporte PNG no Windows apresentava problemas no momento em que escrevíamos este livro.) O único parâmetro que talvez não seja autoexplicativo é o quinto parâmetro para `pdf_place_image( )`. Essa função não está limitada a inserir a imagem em seu tamanho original. O quinto parâmetro é um fator de escola. Escolhemos exibir a imagem em tamanho completo e utilizamos 1 como o fator de escala, mas você poderia utilizar um número maior para aumentar a imagem, ou uma fração para encolhê-la.

O item mais difícil de adicionar ao nosso certificado utilizando PDFlib é a roseta. Não podemos automaticamente abrir e incluir um Windows Meta File contendo a roseta que já temos, mas somos livres para desenhar a forma que quisermos.

Para desenhar uma forma preenchida como um das fitas, podemos escrever o seguinte código. Aqui configuramos o traço ou a cor de linha para ser preta e o preenchimento ou a cor interior para ser azul-marinho:

```
pdf_setcolor($pdf, 'fill', 'rgb', 0, 0, .4, 0);  // azul-escuro
pdf_setcolor($pdf, 'stroke', 'rgb', 0, 0, 0, 0);  // preto
```

Aqui configuramos um polígono de cinco lados para ser uma de nossas fitas e então a preenchemos:

```
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf, 666, 150);
pdf_closepath($pdf);
pdf_fill($pdf);
```

Como também gostaríamos de delinear o polígono, precisamos configurar o mesmo caminho uma segunda vez, mas chamar `pdf_stroke( )` em vez de `pdf_fill( )`.

Como a estrela de múltiplas pontas é uma forma repetitiva e complexa, escrevemos uma função para calcular as localizações no caminho para nós. Nossa função é chamada `draw_star( )` e requer coordenadas x e y para o centro, o número de pontos requeridos, o raio, o comprimento dos pontos, um identificador de documento PDF e um valor booleano para indicar se a forma de estrela deve ser preenchida ou simplesmente contornada.

A função `draw_star( )` utiliza alguns princípios básicos de trigonometria para calcular localizações para uma série de pontos a fim de criar uma estrela. Para cada ponto que solicitamos que a estrela tenha, localizamos um ponto no raio da estrela e um ponto em um círculo menor `$point_size` dentro do círculo externo e desenhamos uma linha entre eles. Algo digno de nota é que funções trigonométricas do PHP como `cos( )` e `sin( )` funcionam em radianos em vez de em graus.

Utilizando uma função e alguma matemática, podemos gerar exatamente uma forma repetitiva e complexa. Se quiséssemos um padrão complicado para nossa borda de página, poderíamos ter utilizado uma abordagem semelhante.

Quando todos os elementos de página forem gerados, precisamos terminar a página e o documento.

Lidando com problemas nos cabeçalhos

Algo menor a notar em todos esses scripts é que precisamos dizer ao navegador o tipo de dados que vamos enviar para ele. Fizemos isso enviando um cabeçalho de HTTP de tipo de conteúdo, por exemplo

```
header( 'Content-type: application/msword' );
```

ou

```
header( 'Content-type: application/pdf' );
```

Devemos estar cientes de que navegadores tratam esses cabeçalhos de forma inconsistente. Em particular, o Internet Explorer costuma ignorar o tipo MIME e tenta automaticamente detectar o tipo de arquivo. (Esse problema particular parece ter melhorado nas versões recentes do Internet Explorer.)

Alguns de nossos cabeçalhos pareceram causar problemas com cabeçalhos de controle de sessão. Há algumas maneiras de contornar isso. Descobrimos que utilizar parâmetros de `GET` em vez de parâmetros de `POST` ou variáveis de sessão evita o problema.

Outra solução é não utilizar um PDF in-line, mas, em vez disso, fazer com que o usuário faça download dele como mostrado no exemplo Hello World PDFlib.

Você também pode evitar problemas se estiver disposto a escrever duas versões ligeiramente diferentes de seu código, uma para o Netscape e uma para o Internet Explorer.

Estendendo o projeto

Adicionar algumas tarefas mais realistas de avaliação para o exame obviamente poderia estender esse projeto, mas ele é realmente projetado como um exemplo de maneiras de entregar seus próprios documentos.

Os documentos personalizados que você talvez queira entregar on-line poderiam incluir documentos legais, formulários de exame ou pedidos parcialmente preenchidos, ou formulários necessários para departamentos do governo.

Leitura adicional

Sugerimos visitar o site da Adobe se você quiser saber mais sobre os formatos PDF (e FDF):

http://www.adobe.com

A seguir

No próximo capítulo, veremos as novas capacidades XML do PHP5 e utilizaremos o PHP para nos conectarmos à API de Web Services da Amazon utilizando REST e SOAP.

33

Conectando-se a serviços Web com XML e SOAP

NOS ÚLTIMOS ANOS, a XML (Extensible Markup Language) tornou-se um meio importante de comunicação. Neste capítulo, utilizaremos a nova interface Web Services da Amazon para construir um carrinho de compras em nosso Web site local que utilize a Amazon como um back-end. (Chamaremos essa aplicação de Tahuayo, que é o nome de um afluente do rio Amazonas.) Utilizaremos dois métodos diferentes para fazer isso: SOAP e REST.REST que também é conhecido como XML sobre HTTP. Utilizaremos a biblioteca interna SimpleXML e a externa NuSOAP para implementar esses dois métodos.

Discutiremos os seguintes tópicos:

- Entendendo os princípios básicos da XML e do SOAP;
- Utilizando a XML para comunicar-se com a Amazon;
- Analisando sintaticamente a XML com a biblioteca de XML do PHP;
- Gravando respostas em cache;
- Conversando com a Amazon com o NuSOAP.

O problema

Temos dois objetivos com este projeto: o primeiro é você compreender o que são a XML e o SOAP e como utilizá-los no PHP. O segundo é colocar essas tecnologias em uso para comunicar-se com o mundo externo. Escolhemos o novo programa Amazon Web Services como um exemplo interessante que você pode achar útil para seu próprio Web site.

A Amazon ofereceu um programa associado que permite anunciar produtos da Amazon em seu Web site. Os usuários podem então seguir um link para cada produto da página no site da Amazon. Se alguém clicar por intermédio de seu site e então comprar esse produto, você obterá uma pequena comissão.

O programa Web Services permite que você use a Amazon mais como um mecanismo: você pode pesquisá-lo e exibir os resultados por meio de seu próprio site ou encher um carrinho de compras do usuário diretamente com o conteúdo de itens que ele selecionou enquanto navegava por seu site. Em outras palavras, o cliente utiliza seu site até o momento de fechar a compra, que ele então pode fazer via Amazon.

A comunicação entre você e a Amazon pode ocorrer de duas maneiras. A primeira é utilizando XML sobre HTTP, o que também é conhecido como REST (Representational State Transfer). Se, por exemplo, você quiser realizar uma busca utilizando esse método, envie uma solicitação HTTP

normal para as informações necessárias, e a Amazon responderá com um documento XML contendo as informações pedidas. Então, você pode analisar esse documento XML e exibir os resultados da busca ao usuário final utilizando uma interface de sua escolha. O processo de enviar e receber dados via HTTP é muito simples, mas a facilidade ou não de analisar o documento resultante depende de sua complexidade.

A segunda maneira é utilizando o SOAP. O *SOAP* é um dos protocolos padrão dos Web Services. Significa Simple Object Access Protocol, mas recentemente foi decidido que o protocolo não era tão simples e que o nome era um pouco equivocado. O resultado é que o protocolo ainda é chamado SOAP mas não é mais um acrônimo.

Neste projeto, construiremos um cliente SOAP que pode enviar solicitações para enviar e receber respostas do servidor SOAP da Amazon. Elas conterão as mesmas informações encontradas nas respostas que obtemos utilizando a XML pelo método de HTTP, mas utilizaremos uma abordagem diferente para extrair os dados, especificamente, a biblioteca de SOAP do PHP.

Nosso objetivo final neste projeto é construir nosso próprio Web site de venda de livros que utilize a Amazon como um back-end. Construiremos duas versões alternativas: uma utilizando a XML pelo HTTP e a outra utilizando o SOAP.

Entendendo XML

Vamos dedicar algum tempo à discussão de XML e Web Services, caso você não esteja familiarizado com esses conceitos.

XML significa Extensible Markup Language. A especificação está disponível do W3C. Uma grande quantidade de informações sobre XML pode ser encontrada no site de XML do W3C em http://www.w3.org/XML/.

A XML é derivada da SGML (Standard Generalized Markup Language). Se você já conhece HTML (e assumimos neste ponto do livro que você provavelmente conheça!), terá pouca dificuldade com os conceitos de XML.

XML é um formato de texto baseado em tags para documentos. Como um exemplo de um documento de XML, a Listagem 33.1 mostra uma das respostas que a Amazon envia a uma solicitação de XML via HTTP.

Listagem 33.1 **Documento de XML descrevendo a primeira edição deste livro**

```
<?xml version="1.0" encoding="UTF-8"?>
<ProductInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation =
                              "http://xml.amazon.com/schemas2/dev-heavy.xsd">
   <Details
url="http://www.amazon.com/exec/obidos/redirect?tag=tangledwebdesign%
26creative=XXXXXXXXXXXXXX%26camp=2025%26
link_code=xm2%26path=ASIN/0672317842">
      <Asin>0672317842</Asin>
      <ProductName>PHP and MySQL Web Development</ProductName>
      <Catalog>Book</Catalog>
      <Authors>
         <Author>Luke Welling</Author>
         <Author>Laura Thomson</Author>
      </Authors>
      <ReleaseDate>30 March, 2001</ReleaseDate>
      <Manufacturer>Sams</Manufacturer>
      <ImageUrlSmall>http://images.amazon.com/images/P/0672317842.01.
THUMBZZZ.jpg</ImageUrlSmall>
      <ImageUrlMedium>http://images.amazon.com/images/P/0672317842.01.
MZZZZZZZ.jpg</ImageUrlMedium>
```

Listagem 33.1 **Continuação**

```
      <ImageUrlLarge>http://images.amazon.com/images/P/0672317842.01.
LZZZZZZZ.jpg</ImageUrlLarge>
      <ListPrice>$49.99</ListPrice>
      <OurPrice>$34.99</OurPrice>
      <UsedPrice>$33.95</UsedPrice>
      <ThirdPartyNewPrice>$33.75</ThirdPartyNewPrice>
      <SalesRank>312</SalesRank>
      <Lists>
         <ListId>3KZW1EV9QMB5F</ListId>
         <ListId>22YC01IGPIZJ3</ListId>
         <ListId>Y2I9B362QXVX</ListId>
      </Lists>
      <BrowseList>
         <BrowseNode>
            <BrowseName>PHP (Computer program language</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>SQL (Computer program language</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Web sites</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Design</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>SQL (Computer language)</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Sql (Programming Language)</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Computer Networks</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Computer Bks - Languages / Programming</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Computers</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Programming Languages - General</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Internet - Web Site Design</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Database Management - SQL Server</BrowseName>
         </BrowseNode>
         <BrowseNode>
            <BrowseName>Programming Languages - SQL</BrowseName>
         </BrowseNode>
      </BrowseList>
      <Media>Paperback</Media>
      <NumMedia>1</NumMedia>
      <Isbn>0672317842</Isbn>
      <Availability>Usually ships within 24 hours</Availability>
      <SimilarProducts>
         <Product>0735709211</Product>
         <Product>1861003730</Product>
```

Listagem 33.1 **Continuação**

```
        <Product>073570970X</Product>
       <Product>1861006918</Product>
        <Product>0596000413</Product>
      </SimilarProducts>
   </Details>
</ProductInfo>
```

O documento começa com a seguinte linha:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Essa é uma declaração padrão que nos informa que o documento que se segue será XML utilizando codificação de caractere UTF-8.

Agora veja o corpo do documento. O documento inteiro consiste em pares de tags de abertura e de fechamento, como:

```
<ProductName>PHP and MySQL Web Development</ProductName>
```

`ProductName` é um elemento, assim como seria em HTML. E, assim como em HTML, podemos aninhar elementos:

```
<Authors>
  <Author>Luke Welling</Author>
  <Author>Laura Thomson</Author>
</Authors>
```

Também como HTML, os elementos podem ter atributos. Por exemplo:

```
<Details url="http://www.amazon.com/exec/obidos/redirect?tag=
➥tangledwebdesign%26creative=XXXXXXXXXXXXXX%26camp=2025%26link_code
➥=xm2%26path=ASIN/0672317842">
```

Esse elemento `Details` possui um único atributo: `url`. Como o URL é muito longo, foi quebrado em linhas aqui.

Há também algumas diferenças de HTML. A primeira é que todas as tags de abertura devem ter uma tag de fechamento. (A exceção para isso é que você pode ter uma única tag se utilizar um formato especial. Se você estiver familiarizado com XHTML, terá visto a tag `<br />` utilizada no lugar de `<br>` por essa exata razão.) Além disso, todos os elementos devem ser aninhados adequadamente. Provavelmente, você sairia com `<b><i>Text</b><</i>` usando um analisador HTML, mas para ser XML ou XHTML, as tags precisam estar aninhadas de forma adequada como `<b><i>Text</i></b>`.

A principal diferença que você notará entre XML e HTML é que parecemos estar compondo nossas próprias tags como veremos em seguida! Essa é a flexibilidade da XML. Podemos estruturar nossos documentos para corresponder aos dados que queremos armazenar. Podemos formalizar a estrutura de documentos de XML escrevendo uma DTD (Document Type Definition) ou XML Schema. Esses documentos são utilizados para descrever a estrutura de um dado documento de XML. Se você quiser, uma DTD ou Schema é como uma declaração de classe; e o documento de XML é como uma instância dessa classe. Neste exemplo em particular, não usamos nem DTD nem Schema.

Você pode ler a DTD da Amazon para esse documento em:

http://xml.amazon.com/schemas2/dev-heavy.dtd

Você pode ler o XML Schema em:

http://xml.amazon.com/schemas2/dev-heavy.xsd

Você não será capaz de abrir o arquivo de DTD em alguns navegadores, porque eles tentarão analisar sintaticamente o DTD como XML e ficarão confusos. Você pode, porém, fazer o download e lê-lo no editor de sua escolha. Você deve ser capaz de abrir o XML Schema diretamente no seu navegador.

Você notará que, além da declaração inicial de XML, o corpo inteiro do documento está contido dentro do elemento ProductInfo. Isso é chamado *elemento raiz* do documento. Vamos fazer um exame mais minucioso:

```
<ProductInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="http://xml.amazon.com/schemas2/
     dev-heavy.xsd">
```

Você verá que tem alguns atributos ligeiramente incomuns. Esses são os *espaços de nome XML*. Não precisamos entender espaços de nome para o que queremos fazer neste projeto, mas eles podem ser muito úteis. A idéia básica é qualificar o elemento e atribuir nomes com um espaço de nome de modo que os nomes comuns não entrem em conflito quando lidarem com documentos de fontes diferentes.

Se quiser saber mais sobre espaços de nome, você pode ler o documento "Namespaces in XML Recommendation" em http://www.w3.org/TR/REC-xml-names/.

Se você quiser saber mais sobre XML em geral, há uma grande variedade de recursos. O site do W3C é um excelente lugar para começar, e também há literalmente centenas de livros excelentes e tutoriais Web. ZVON.org é um dos melhores na Web.

Entendendo os serviços Web

Os serviços Web são interfaces de aplicação que se tornaram disponíveis na World Wide Web. Se você quiser, um serviço Web pode ser visto como uma classe que expõe seus métodos públicos via Web. Os serviços Web agora estão tornando-se amplamente difundidos e alguns dos maiores nomes em negócios estão tornando algumas de suas funcionalidades disponíveis via serviços Web.

Por exemplo, o Google, Amazon, eBay e PayPal agora oferecem uma variedade de serviços Web. Depois que tiver passado pelo processo de configuração de um cliente para a interface da Amazon neste capítulo, você deverá achar muito simples construir uma interface de cliente para o Google. Você pode obter informações adicionais em http://www.google.com/apis/.

Uma lista de serviços Web públicos que está sempre crescendo está disponível em http://www.xmethods.net.

Há vários protocolos de núcleo envolvidos nessa metodologia de chamada remota de função. Dois dos mais importantes são SOAP e WSDL.

SOAP

O SOAP é um protocolo de troca de mensagens controlado por solicitação e resposta que permite aos clientes chamarem serviços Web, e permite aos servidores responderem. Cada mensagem de SOAP, uma solicitação ou uma resposta, é um documento simples de XML. Uma solicitação de SOAP de exemplo que poderíamos enviar para a Amazon é mostrada na Listagem 33.2.

Listagem 33.2 **Solicitação de SOAP para uma pesquisa baseada em ASIN**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
➥envelope/"
                 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/
➥encoding/"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
   <namespl:AsinSearchRequest xmlns:namespl="urn:PI/DevCentral/
➥SoapService">
     <AsinSearchRequest xsi:type="m:AsinRequest">
```

Listagem 33.2 Continuação

```
        <asin >0060518057</asin>
        <tag >your-associate-id</tag>
        <type >heavy</type>
        <dev-tag >your-dev-tag</dev-tag>
      </AsinSearchRequest>
    </namespl:AsinSearchRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A mensagem de SOAP inicia com a declaração de que isso é um documento de XML. O elemento raiz de todas as mensagens de SOAP é o envelope de SOAP. Dentro dele localizamos o elemento `Body` que contém a solicitação real.

Essa solicitação é um `AsinSearchRequest`, que pede ao servidor da Amazon que pesquise um item particular em seu banco de dados com base no ASIN, que significa Amazon.com Standard Item Number. Esse é um identificador único dado a cada produto no banco de dados da Amazon.com.

Pense no `AsinSearchRequest` como uma chamada de função em uma máquina remota, e nos elementos contidos dentro desse elemento, como os parâmetros que estamos passando a essa função. Nesse caso, estamos passando um ASIN para o livro de Dilbert *Way of the Weasel*. Também precisamos passar na tag, que é seu Associate ID, o tipo de pesquisa a executar (`heavy` ou `lite`) e a `dev-tag`, que é um valor de token de desenvolvedor que a Amazon lhe fornecerá. O tipo de elemento informa ao serviço se queremos detalhe limitado (`lite`) ou todas as informações disponíveis (`heavy`).

A resposta para essa solicitação é muito semelhante ao documento de XML que vimos na Listagem 33.1, mas incluído em um envelope de SOAP.

Quando lida com SOAP, em geral você gera solicitações de SOAP e interpreta as respostas programaticamente utilizando uma biblioteca de SOAP, independente da linguagem de programação que estiver utilizando. Isso é bom pois poupa o esforço de construir a solicitação de SOAP e interpretar a resposta manualmente.

WSDL

WSDL significa *Web Services Description Language*. (Freqüentemente pronunciado como "wiz-dul".) É utilizado para descrever a interface para serviços disponíveis em um Web site particular. Se você quiser ver o documento de WSDL que descreve o Amazon Web Services que usaremos neste capítulo, ele está localizado em http://soap.amazon.com/schemas2/AmazonWebServices.wsdl.

Como verá se seguir esse link, os documentos de WSDL são significativamente mais complexos do que as mensagens de SOAP. Você sempre os geraria e interpretaria programaticamente, se fosse dada uma escolha.

Se quiser saber mais sobre WSDL, você pode consultar o W3C Draft em:
http://www.w3.org/TR/wsd20/.

No momento em que escrevíamos este livro, a WSDL não era ainda uma recomendação do W3C; portanto, é ainda um assunto a ser alterado. Isso não impediu os desenvolvedores de a utilizarem entusiasticamente. Entretanto, como todas as peças do quebra-cabeça de serviços Web, ele está sujeito a mudanças já que toda a área está se desenvolvendo tão rápido.

Componentes da solução

Existem algumas partes necessárias para construir a solução. Assim como aquelas mais óbvias – uma interface de carrinho de compras para exibir aos clientes, e código para conectar-se à Amazon via REST ou SOAP – você precisa de algumas partes auxiliares. Depois de recuperado o documento XML, o código precisa analisá-lo a fim de extrair as informações que o carrinho exibirá. Para atender às exigências da Amazon e melhorar o desempenho, é preciso considerar armazenamento em

cache. Finalmente, como a atividade de saída precisa ser feita na Amazon, você precisa de uma funcionalidade para entregar o conteúdo do carrinho do usuário para a Amazon e passar o usuário para aquele serviço.

Construindo um carrinho de compras

Obviamente precisaremos construir um carrinho de compras como o front-end para o sistema. Fizemos isso antes, no Capítulo 27. Como os carrinhos de compras não são o principal foco deste projeto, utilizaremos uma aplicação simplificada. Só precisamos fornecer um carrinho básico de modo que possamos monitorar o que o cliente gostaria de comprar e informar à Amazon após o checkout, isto é, a passagem pelo caixa.

Usando as interfaces de serviços Web da Amazon

Para utilizar a interface Web Services da Amazon, você precisa fazer download do Amazon Web Services Developers' Kit. Nós o obtivemos em http://www.amazon.com/gp/aws/landing.html. No entanto, esse URL está sujeito a mudanças.

Você também precisará se inscrever para um Developer Token. Você pode fazer isso no mesmo site. Esse token é utilizado para identificá-lo na Amazon quando suas solicitações chegarem.

Talvez você queira se inscrever para um ID de associado da Amazon. Isso permitirá que você ganhe comissão se as pessoas comprarem qualquer produto via sua interface.

Quando fizer o download do kit de desenvolvedor, leia-o por inteiro. Ele vem com a documentação sobre como a interface funciona e amostras de código em várias linguagens, incluindo PHP.

Antes que possa fazer o download, você precisa concordar com o acordo de licenciamento. Vale a pena ler, pois não é o blablablá de licenciamento de software usual. Algumas condições de licenciamento que são importantes durante a implementação são:

- Você não deve fazer mais de uma solicitação por segundo.
- Você deve gravar em cache os dados que vêm da Amazon.
- Se você gravar os preços e disponibilidade em cache por mais de uma hora, deve renunciar a seus direitos.
- Você deve gravar a maior parte dos dados em cache por 24 horas e alguns atributos estáveis por até três meses.
- Você deve vincular para uma página na Amazon.com e não pode criar links com texto ou figura dos quais fez download da Amazon para outro Web site comercial.

Com um nome de domínio difícil de soletrar, nenhuma promoção e nenhuma razão óbvia para utilizar Tahuayo.com em vez de ir diretamente à Amazon.com, não precisamos dar nenhum passo adicional para manter as solicitações abaixo de uma por segundo.

Implementamos a gravação em cache para atender às condições nos pontos 2 a 4. Gravamos imagens em cache por 24 horas; e os dados de produto (que contêm preços), por uma hora.

Sua aplicação também segue o quinto ponto. Você deseja que os itens na página principal tenham links a páginas detalhadas no site, mas estabele o link com a Amazon quando o pedido está completo.

Análise sintática de XML

A primeira interface que a Amazon oferece a seus Web Services é via REST. Essa interface aceita uma solicitação http normal e retorna um documento XML. Para utilizar essa interface, é preciso que você possa analisar a resposta XML que a Amazon envia de volta para você. Isso pode ser feito utilizando a biblioteca SimpleXML do PHP. Essa biblioteca requer pelo menos a versão 5.0.0 do PHP, mas está ativada por padrão.

Usando SOAP com PHP

A outra interface que oferece os mesmos Web Services é SOAP. Para acessar esses serviços utilizando SOAP, você precisa utilizar um das diversas bibliotecas SOAP do PHP. Existe uma biblioteca SOAP embutida, mas como nem sempre está disponível, você pode utilizar a biblioteca NuSOAP. Como NuSOAP é escrita em PHP, não precisa de compilação. É apenas um único arquivo a ser chamado via `require_once( )`.

NuSOAP está disponível em http://dietrich.ganx4.com/nusoap/. E está disponível sob Lesser GPL; ou seja, você pode utilizá-la em qualquer aplicação, incluindo aplicações não-livres.

Gravando em cache

Como mencionamos antes, um dos termos e condições impostos aos desenvolvedores pela Amazon é que os dados descarregados da Amazon via serviços Web devem ser gravados em cache. Em nossa solução, precisaremos encontrar uma forma de armazenar e reutilizar os dados descarregados até que tenha passado a data de validade.

Visão geral da solução

Para este projeto, utilizaremos novamente uma abordagem baseada em evento para escrever nosso código, como fizemos nos Capítulos 29 e 30. Não desenharemos um diagrama de fluxo de sistema neste exemplo, pois há apenas algumas telas no sistema e os links entre elas são simples.

Os usuários começarão na tela principal do Tahuayo, mostrado na Figura 33.1.

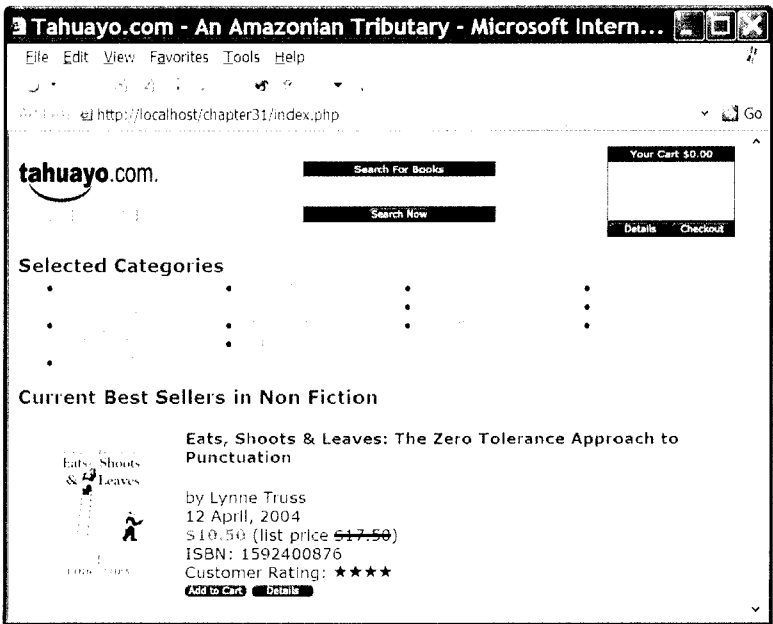



Figura 33.1 A primeira tela para o Tahuayo mostra todos os recursos principais do site: navegação por categoria, pesquisa e o carrinho de compras.

Como você pode ver, os recursos principais do site são a exibição de Selected Category e os itens nessas categorias. Por padrão, estamos exibindo a categoria Current Best Sellers na página frontal. Se um usuário clicar em outra categoria, ele verá uma exibição semelhante a essa categoria.

Uma breve explicação da terminologia antes de avançarmos mais: a Amazon se refere às categorias como *nós de navegação (browse nodes)*. Você verá essa expressão utilizada em todo nosso código e na documentação oficial.

A documentação fornece uma lista parcial de nós de navegação populares. Além disso, se quiser um nó particular, você pode navegar pelo site normal da Amazon.com e lê-lo a partir do URL, mas não há nenhuma maneira de obter uma lista completa. Infelizmente, algumas categorias importantes, como os livros mais vendidos, não podem ser acessadas como nós de navegação.

Note que há mais livros e links para páginas adicionais na parte inferior dessa página que você não pode ver na captura de tela. Estamos exibindo 10 livros em cada página, junto com links para até 30 outras páginas. Esse valor de 10 por página é definido pela Amazon. O limite de 30 páginas é nossa própria escolha arbitrária.

A partir daqui, os usuários podem seguir links para informações detalhadas sobre livros individuais. Essa tela é mostrada na Figura 33.2.

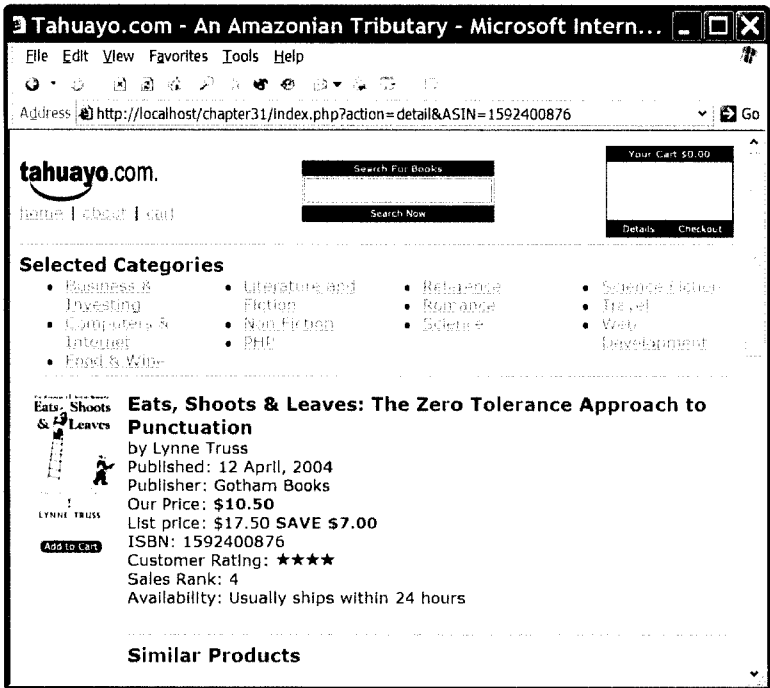



Figura 33.2 Nossa página de detalhes mostra as informações adicionais sobre um livro particular, incluindo produtos semelhantes e críticas.

Apesar de não caber em uma captura de tela, estamos mostrando nessa página a maioria das, mas não todas as, informações que a Amazon envia com uma solicitação pesada. Estamos optando por ignorar partes destinadas a outros produtos além dos livros e a lista de categorias em que o livro se encaixa.

Se o usuário clicar na imagem da capa, ele poderá ver uma versão maior da imagem.

Você deve ter notado a caixa de pesquisa na parte superior da tela nessas figuras. Essa pesquisa executará uma busca de palavras-chave no catálogo da Amazon usando a interface Web Services do nosso site. Um exemplo da saída de uma pesquisa é mostrado na Figura 33.3.

Embora tenhamos listado apenas algumas categorias, os clientes podem obter qualquer livro por meio do recurso de pesquisa e da navegação para livros particulares.

Cada livro individual tem um link Add to Cart com ele. Clicando nesse ou no link Details no resumo do carrinho somos levados a uma exibição do conteúdo do carrinho. Isso é mostrado na Figura 33.4.

Por fim, quando um cliente conclui a compra clicando em um dos links Checkout, enviamos os detalhes do seu carrinho de compras para a Amazon e o levamos até lá. Ele verá uma página semelhante a essa na Figura 33.5.

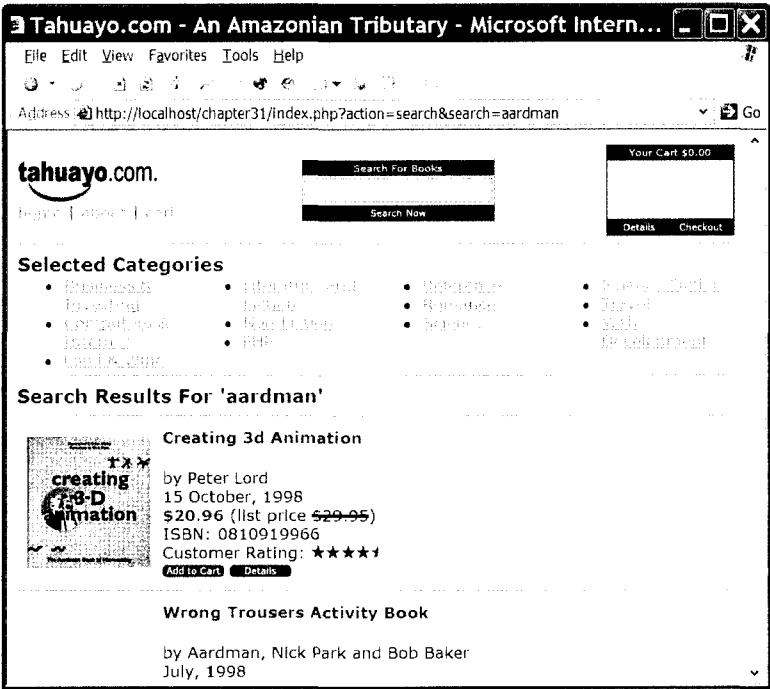



Figura 33.3 Os resultados da pesquisa para aardman.

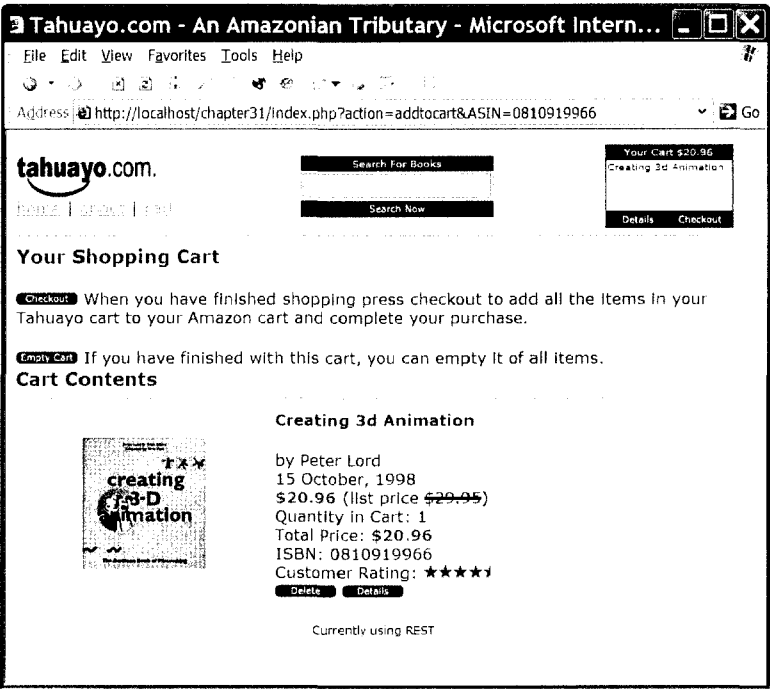



Figura 33.4 A partir da página de carrinho de compras, podemos excluir itens, limpar o carrinho ou concluir a compra.

Você agora deve entender o que queremos dizer ao construir nosso próprio front-end e ao usar a Amazon como back-end.

Como novamente usamos a abordagem baseada em evento, a parte principal da decisão que tem a lógica da aplicação está em um arquivo, `index.php`. Uma visão geral dos arquivos na aplicação é mostrada na Tabela 33.1.

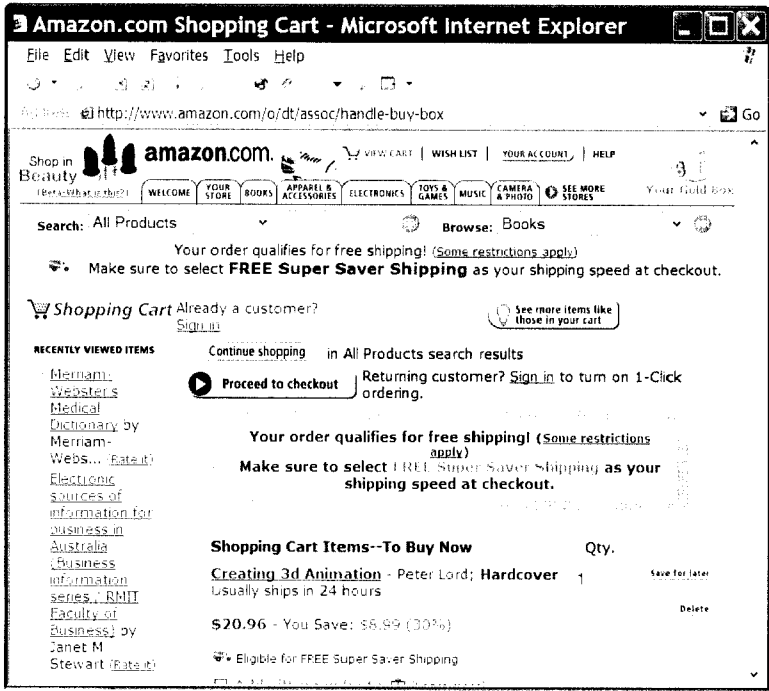



Figura 33.5 Os itens que estavam no carrinho de Tahuayo do cliente estão agora em seu carrinho da Amazon.

Tabela 33.1 **Arquivos na aplicação Tahuayo**

Nome de arquivo	Tipo	Descrição
`index.php`	Aplicação	O principal arquivo da aplicação.
`about.php`	Aplicação	Mostra a página About.
`constants.php`	Arquivo Include	Configura algumas constantes globais.
`topbar.php`	Arquivo Include	Gera a barra de informações e a CSS na parte superior de cada página.
`bottom.php`	Arquivo Include	Gera o rodapé na parte inferior de cada página.
`AmazonResultSet.php`	Arquivo de classe	Contém a classe do PHP que armazena o resultado de cada consulta da Amazon.
`Product.php`	Arquivo de classe	Contém a classe do PHP que armazena as informações sobre um livro particular.
`bookdisplayfunctions.php`	Funções	Contém as funções que ajudam a exibir um livro e listas de livros.
`cachefunctions.php`	Funções	Contém funções para executar o cache requerido pela Amazon.
`cartfunctions.php`	Funções	Contém funções relacionadas ao carrinho de compras.
`categoryfunctions.php`	Funções	Contém funções que ajudam a recuperar e a exibir uma categoria.
`utilityfunctions.php`	Funções	Contém um conjunto de funções utilitárias utilizadas por toda a aplicação.

Você também precisará do arquivo nusoap.php que mencionamos anteriormente, como é exigido nesses arquivos. O NuSOAP está no diretório chapter33 no CD-ROM que acompanha este livro, mas talvez você queira substituí-lo por uma versão mais nova em http://dietrich.ganx4. com/nusoap/index.php.

Começaremos verificando o arquivo de aplicação principal index.php.

Aplicação principal

O arquivo de aplicação index.php é mostrado na Listagem 33.3.

Listagem 33.3 **index.php – O arquivo da aplicação principal**

```
<?php
//Estamos apenas utilizando uma variável de sessão 'cart' para armazenar o conteúdo do carrinho
session_start( );

require_once('constants.php');
require_once('Product.php');
require_once('AmazonResultSet.php');
require_once('utilityfunctions.php');
require_once('bookdisplayfunctions.php');
require_once('cartfunctions.php');
require_once('categoryfunctions.php');

// Essas são as variáveis que estamos esperando de fora.
// Elas serão validadas e convertidas em globais
$external = array('action', 'ASIN', 'mode', 'browseNode', 'page', 'search');

// as variáveis podem vir via Get ou Post
// converte todas as variáveis externas esperadas em nomes globais abreviados
foreach ($external as $e)
{
 if(@$_REQUEST[$e])
   $$e = $_REQUEST[$e];
 else
   $$e = '';

 $$e = trim($$e);
}

// valores padrão para variáveis globais
if($mode=='')
 $mode = 'books'; // Nenhum outro modo foi testado
if($browseNode=='')
 $browseNode = 53; //53 são os livrso de não-ficção mais bem vendidos
if($page=='')
 $page = 1;  // Primeira página – há 10 itens por página

//validate/strip input
if(!eregi('^[A-Z0-9]+$', $ASIN)) // ASINS deve ser alfanumérico
 $ASIN ='';
if(!eregi('^[a-z]+$', $mode)) // modo deve ser alfabético
 $mode = 'books';
$page=intval($page); // páginas e browseNodes devem ser inteiros
$browseNode = intval($browseNode);
// pode gerar alguma confusão, mas estamos retirando os caracteres
// $search só parece ser razoável modificá-lo agora para que seja exibido
// no cabeçalho
$search = safeString($search);
```

Listagem 33.3 **Continuação**

```
if(!isset($_SESSION['cart']))
{
 session_register('cart');
 $_SESSION['cart'] = array( );
}

// tarefas que precisam ser feitas antes da barra superior ser exibida
if($action == 'addtocart')
 addToCart($_SESSION['cart'], $ASIN, $mode);
if($action == 'deletefromcart')
 deleteFromCart($_SESSION['cart'], $ASIN) ;
if($action == 'emptycart')
 $_SESSION['cart'] = array( );
// mostra a barra superior
require_once ('topbar.php');

// loop de evento principal. Reage à ação do usuário na página de chamada
switch ($action)
{
 case 'detail' :
   showCategories($mode);
   showDetail($ASIN, $mode);
 break;

 case 'addtocart' :
 case 'deletefromcart' :
 case 'emptycart' :
 case 'showcart' :
   echo '<hr /><h1>Your Shopping Cart</h1>';
   showCart($_SESSION['cart'], $mode);
 break;

 case 'image' :
   showCategories($mode);
   echo '<h1>Large Product Image</h1>';
   showImage($ASIN, $mode);
 break;

 case 'search' :
   showCategories($mode);
   echo "<h1>Search Results For '$search'</h1>";
   showSearch($search, $page, $mode);
 break;

 case 'browsenode':
 default:
   showCategories($mode);
   $category = getCategoryName($browseNode);
   if(!$category||$category=='Best Selling Books')
   {
     echo '<h1>Current Best Sellers</h1>';
   }
   else
   {
     echo "<h1>Current Best Sellers in $category</h1>";
   }
   showBrowseNode($browseNode, $page, $mode) ;
 break;
}
require ('bottom.php');
?>
```

Vamos examinar esse arquivo. Iniciamos criando uma sessão. Armazenaremos o carrinho de compras do cliente como uma variável de sessão como fizemos antes.

Então, incluímos vários arquivos. A maioria deles são funções que discutiremos mais tarde, mas precisamos conversar sobre o primeiro arquivo incluído agora. Esse arquivo, `constants.php`, define algumas constantes importantes que serão utilizadas por toda a aplicação. O conteúdo de `constants.php` pode ser localizado na Listagem 33.4.

Listagem 33.4 **constants.php – Declaração de constantes de globais de chave**

```
<?php
// essa aplicação pode se conectar via XML HTTP ou SOAP
// Define uma versão de METHOD a escolher.
define('METHOD', 'SOAP');
//define('METHOD', 'REST');

// certifique-se de criar um diretório de cache e torná-lo gravável
define('CACHE', 'cache'); // caminho para arquivos gravados em cache
define('ASSOCIATEID', 'webservices-20'); //coloque seu Associate ID aqui
define('DEVTAG', 'XXXXXXXXXXXXXX'); // coloque sua tag de desenvolvedor aqui
//fornece um erro se o software for executado com o devtag fictício
if(DEVTAG=='XXXXXXXXXXXXXX')
{
  die ('You need to sign up for an Amazon.com developer tag at<a href =
    "https://associates.amazon.com/exec/panama/associates/join/
➥developer/application.html/ref=sc_bb_1_0/002">Amazon</a>
      when you install this software.  You should probably sign up
      for an associate ID at the same time. Edit the file constants.php.');
}

// (parcial) lista de nós de navegação da Amazon.
$categoryList = array(5=>'Computers & Internet', 3510=>'Web Development',
                      295223=>'PHP', 17=>'Literature and Fiction',
                      3=>'Business & Investing', 53=>'Non Fiction',
                      23=>'Romance', 75=>'Science', 21=>'Reference',
                      6 =>'Food & Wine', 27=>'Travel',
                      16272=>'Science Fiction'
                     );
?>
```

Essa aplicação foi desenvolvida para utilizar REST ou SOAP. Você pode configurar qual deve ser usado mudando o valor da constante `METHOD`.

A constante `CACHE` mantém o caminho para o cache para os dados descarregados da Amazon. Altere para o caminho que você gostaria de utilizar em seu sistema.

A constante `ASSOCIATEID` mantém o valor de seu Associate ID. Se você enviar isso para a Amazon com transações, obterá uma comissão. Altere isso para seu próprio Associate ID.

A constante `DEVTAG` mantém o valor do token que a Amazon lhe fornecerá quando você se inscrever. Você precisa alterar isso para sua própria tag de desenvolvedor ou a aplicação não funcionará. Você pode se inscrever para uma tag em:

http://www.amazon.com/gp/aws/landing.html

Vamos examinar novamente `index.php`. Ele contém algumas preliminares e o loop de evento principal. Começamos extraindo quaisquer variáveis entrantes que cheguem via `GET` ou `POST`. Em seguida, configuramos alguns valores padrão para algumas variáveis globais padrão que determinam o que será exibido mais tarde, como segue:

```
// valores padrão para variáveis globais
if($mode=='')
  $mode = 'books'; // Nenhum outro modo foi testado
```

```
if($browseNode=='')
  $browseNode = 53; //53 é o valor para livros de não-ficção
if($page=='')
  $page = 1; // Primeira Página - há 10 itens por página
```

Configuramos a variável mode para books. A Amazon suporta muitos outros modos (tipos de produtos), mas para essa aplicação, nos preocuparemos apenas com os livros. Não deve ser muito difícil modificar o código neste capítulo para lidar com outras categorias. O primeiro passo nessa expansão seria redefinir $mode. Você precisaria verificar a documentação da Amazon para ver quais outros atributos serão retornados para produtos que não sejam livros e remover a linguagem específica de livro da interface com o usuário.

A variável browseNode é utilizada para especificar qual categoria de livros gostaríamos de exibir. Isso pode ser configurado se o usuário tiver clicado em um dos links Selected Categories. Se não estiver configurado – por exemplo, quando o usuário entra no site pela primeira vez – configuraremos para 53. Os nós de navegação da Amazon são simplesmente inteiros que identificam uma categoria. O valor 53 representa a categoria Livros de Não-Ficção, que parece um bom nó como qualquer outro para exibir na página inicial, dado que algumas das melhores categorias genéricas não estão disponíveis como nós de navegação.

A variável page é utilizada para dizer à Amazon que subconjunto de resultados gostaríamos de ter exibido dentro de uma dada categoria. A página 1 contém resultados 1–10, a página 2 tem resultados 11–20 e assim por diante. A Amazon configura o número de itens em uma página e não temos controle sobre isso. Naturalmente, poderíamos exibir duas ou mais "páginas" de dados da Amazon em uma de nossas páginas, mas 10 é um número razoável e o caminho de menor resistência.

Em seguida, organizamos quaisquer dados de entrada que recebemos, seja pela caixa de pesquisa ou via parâmetros GET ou POST:

```
//validate/strip input
if(!eregi('^[A-Z0-9]+$', $ASIN)) // ASINS deve ser alfanumérico
  $ASIN ='';
if(!eregi('^[a-z]+$', $mode)) // modo deve ser alfabético
  $mode = 'books';
$page=intval($page); // páginas e browseNodes devem ser inteiros
$browseNode = intval($browseNode);
// pode causar alguma confusão, mas estamos eliminando alguns caracteres
// de $search
// só parece razoável modificá-lo agora para ele ser exibido no
// título
$search = safeString($search) ;
```

Isso não é nada novo. A função safeString( ) está no arquivo de utilityfunctions.php. Ela simplesmente remove qualquer caractere não-alfanumérico da string de entrada via uma substituição de expressão regular. Como abordamos isso antes, não incluímos aqui no texto.

A razão principal por que precisamos validar a entrada nessa aplicação é que utilizamos a entrada do cliente para criar nomes de arquivo no cache. Poderíamos incorrer em problemas sérios se permitíssemos aos clientes incluírem .. ou / em sua entrada.

Em seguida, configuramos o carrinho de compras do cliente, se ele ainda não tiver um:

```
if(!isset($_SESSION['cart']))
{
 session_register('cart');
 $_SESSION['cart'] = array( );
}
```

Ainda temos algumas últimas coisas a fazer antes de poder exibir as informações na barra superior de informações na página (veja a Figura 33.1 para se lembrar da aparência que isso tem). Um resumo do carrinho de compras é mostrado na barra superior de cada página. Portanto, é importante que a variável cart seja atualizada antes de isso ser exibido.

```
// tarefas que precisam ser feitas antes de a barra superior ser exibida
if($action == 'addtocart')
 addToCart($_SESSION['cart'], $ASIN, $mode);
if($action == 'deletefromcart')
 deleteFromCart($_SESSION['cart'], $ASIN) ;
if($action == 'emptycart')
 $_SESSION['cart'] = array( );
```

Aqui estamos adicionando ou excluindo itens do carrinho conforme necessário antes de exibir o carrinho. Voltaremos a essas funções quando discutirmos o carrinho de compras e concluirmos a compra. Se quiser vê-los agora, eles estão no arquivo `cartfunctions.php`. Vamos deixá-los de lado por um minuto, porque precisamos entender a interface para a Amazon primeiro.

Em seguida, incluímos o arquivo `topbar.php`. Esse arquivo simplesmente contém HTML, uma folha de estilos e uma única chamada de função para a função `ShowSmallCart( )` (de `cartfunctions.php`). Isso exibe o pequeno resumo do carrinho de compras que você pode ver no canto direito superior das figuras. Voltaremos para esse assunto quando discutirmos as funções de cart.

Por fim, chegamos ao loop de tratamento de evento principal. Um resumo das possíveis ações é mostrado na Tabela 33.2.

Tabela 33.2 **Possíveis ações no loop de evento principal**

Ação	Descrição
`browsenode`	Mostra os livros na categoria especificada. Essa é a ação padrão.
`detail`	Mostra os detalhes de um livro particular.
`image`	Mostra uma versão grande da capa do livro.
`search`	Mostra os resultados de uma pesquisa de usuário.
`addtocart`	Adiciona um item ao carrinho de compras do usuário.
`deletefromcart`	Exclui um item do carrinho de compras.
`emptycart`	Esvazia todo o carrinho de compras.
`Showcart`	Mostra o conteúdo do carrinho.

Como você pode ver, as primeiras quatro ações nessa tabela se referem à recuperação e à exibição de informações a partir da Amazon. O segundo grupo de quatro lida com gerenciamento do carrinho de compras.

As ações que recuperam dados a partir da Amazon funcionam de maneira similar. Consideraremos a recuperação de dados sobre livros em um `browsenode` (categoria) particular como um exemplo.

Mostrando livros em uma categoria

O código que é executado quando a ação é `browsenode` (visualizar uma categoria) é o seguinte:

```
showCategories($mode);
$category = getCategoryName($browseNode);
if(!$category||$category=='Best Selling Books')
{
  echo '<h1>Current Best Sellers</h1>';
}
else
{
  echo "<h1>Current Best Sellers in $category</h1>";
}
showBrowseNode($browseNode, $page, $mode);
```

A função showCategories( ) exibe a lista de categorias selecionadas que vemos próximo à parte superior da maioria das páginas. A função getCategoryName( ) retorna o nome da categoria atual dado seu número de browsenode. A função showBrowseNode( ) exibe uma página de livros nessa categoria.

Vamos começar considerando a função showCategories( ). O código para essa função é mostrado na Listagem 33.5.

Listagem 33.5 A função **showCategories( )** de **categoryfunctions.php** – Uma lista de categorias

```
//exibe uma lista inicial de categorias populares
function showCategories($mode)
{
  global $categoryList;
  echo '<hr /><h2>Selected Categories</h2>';

  if($mode == 'books')
  {

    asort($categoryList);

    $categories = count($categoryList);
   $columns = 4;
    $rows = ceil($categories/$columns);

    echo '<table border ="0" cellpadding ="0" cellspacing="0"
                width ="100%"><tr>';

    reset($categoryList);

    for($col = 0; $col<$columns; $col++)
    {
      echo '<td width = "'.(100/$columns).'%" valign = "top"><ul>';
      for($row = 0; $row<$rows; $row++)
      {
        $category = each($categoryList);
        if($category)
        {
          $browseNode = $category['key'];
          $name = $category['value'];
          echo "<li><span class = 'category'><a href =
                'index.php?action=browsnode&browseNode=$browseNode'>$name
                </a></span></li>";
        }
      }
      echo '</ul></td>';
    }
    echo '</tr></table><hr />';
  }
}
```

Essa função utiliza um array chamado categoryList, declarado no pacote categoryfunctions.php, para mapear os números de browsenode para nomes. Os browsenodes desejados são simplesmente codificados diretamente nesse array. Essa função classifica o array e exibe as várias categorias.

A função getCategoryName( ), que é chamada em seguida no loop do evento principal, é utilizada para pesquisar o nome do browsenode que estamos vendo atualmente de modo que possamos exibir um título na tela como Current Best Sellers in Business & Investing. Ela procura isso no array categoryList mencionado anteriormente.

A diversão começa realmente quando obtemos a função showBrowseNode( ). Essa função é mostrada na Listagem 33.6.

Listagem 33.6 **Função showBrowseNode( ) de bookdisplayfunctions.php –** Uma lista de categorias

```
// Para um browsenode particular, exibe uma página de produtos
function showBrowseNode($browseNode, $page, $mode)
{
  $ars = getARS('browse', array('browsenode'=>$browseNode,
                                'page' => $page,
                                'mode'=>$mode));
  showSummary($ars->products( ), $page, $ars->totalResults( ),
              $mode, $browseNode);
}
```

Essa função faz exatamente duas coisas. Primeiro, chama a função getARS( ) de cachefunctions.php. Essa função obtém e retorna um objeto de AmazonResultSet (saberemos mais sobre isso a seguir). Em seguida chama a função showSummary( ) de bookdisplayfunctions.php para exibir as informações recuperadas.

A função getARS( ) é absolutamente fundamental para guiar a aplicação inteira. Se examinarmos o código para as outras ações – exibir detalhes, imagens e pesquisa – descobriremos que tudo volta para isso.

Obtendo uma classe AmazonResultSet

Vejamos essa função getARS( ) em mais detalhe. Esse código é mostrado na Listagem 33.7.

Listagem 33.7 **Função getARS( ) de cachefunctions.php – Um conjunto de resultados para uma consulta**

```
// Obtém um AmazonResultSet do cache ou de uma consulta ao vivo
// Se uma consulta ao vivo o adicionar ao cache
function getARS($type, $parameters)
{
  $cache = cached($type, $parameters);
  if($cache) // se encontrado no cache
  {
    return  $cache;
  }
  else
  {
    $ars = new AmazonResultSet;
    if($type == 'asin')
      $ars->ASINSearch(padASIN($parameters['asin']), $parameters['mode']);
    if($type == 'browse')
      $ars->browseNodeSearch($parameters['browsenode'],
$parameters['page'], $parameters['mode']);
    if($type == 'search')
      $ars->keywordSearch($parameters['search'], $parameters['page'],
                          $parameters['mode']);
    cache($type, $parameters, $ars);
  }
  return $ars;
}
```

Essa função foi projetada para controlar o processo de obtenção de dados da Amazon. Você pode fazer isso de duas maneiras: a partir do cache ou a partir da Amazon. Como a Amazon exige que os desenvolvedores gravem em cache os dados descarregados, a função primeiro procura os dados no cache. Discutiremos o cache em algumas páginas.

Se ainda não realizamos essa consulta particular, os dados devem ser buscados ao vivo na Amazon. Fazemos isso criando uma instância da classe AmazonResultSet e chamando o método que corresponde à consulta particular que queremos executar. O tipo de consulta é determinado pelo parâmetro $type. No exemplo de pesquisa de categoria (ou nó de navegação), passamos browse como o valor para esse parâmetro – consulte a Listagem 33.6. Se quisermos realizar uma consulta sobre um livro particular, devemos passar o valor asin e se quisermos realizar uma pesquisa de palavra-chave, o parâmetro deve ser configurado como search.

Cada um desses parâmetros chama um método diferente na classe AmazonResultSet. A pesquisa de item individual chama o método ASINSearch( ). A pesquisa de categoria chama o método browseNodeSearch( ). A pesquisa de palavra-chave chama o método keywordSearch( ).

Vejamos mais de perto a classe AmazonResultSet. O código completo para essa classe é mostrado na Listagem 33.8.

Listagem 33.8 **Amazonresultset.php – Uma classe para tratar conexões da Amazon**

```
<?php
// você pode alternar entre REST e SOAP utilizando esta constante definida em
// constants.php
if(METHOD=='SOAP')
{
 include_once('nusoap/nusoap.php');
}

// Essa classe armazena o resultado das consultas
// Geralmente são 1 ou 10 instâncias da classe Product
class AmazonResultSet
{
 private $browseNode;
 private $page;
 private $mode;
 private $url;
 private $type;
 private $totalResults;
 private $currentProduct = null;
 private $products = array( ); // array de objetos Product

 function products( )
 {
   return $this->products;
 }

 function totalResults( )
 {
   return $this->totalResults;
 }

 function getProduct($i)
 {
   if(isset($this->products[$i]))
     return $this->products[$i] ;
   else
     return false;
 }
```

Listagem 33.8 **Continuação**

```
// Realiza uma consulta para obter uma página cheia de produtos a partir de um nó de navegação
// Alterna entre XML/HTTP e SOAP em constants.php
// Retorna um array de Products
function browseNodeSearch($browseNode, $page, $mode)
{
  if(METHOD=='SOAP')
  {
    $soapclient = new soapclient (
          'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
          'wsdl');
    $soap_proxy = $soapclient->getProxy( );
    $parameters['mode']=$mode;
    $parameters['page']=$page;
    $parameters['type']='heavy';
    $parameters['tag']=$this->assocID;
    $parameters['devtag']=$this->devTag;
    $parameters['sort']='+salesrank';
    $parameters['browse_node'] = $browseNode;

    // realiza consulta soap real
    $result = $soap_proxy->BrowseNodeSearchRequest($parameters);
    if(isSOAPError($result))
      return false;
    $this->totalResults = $result['TotalResults'];

    foreach($result['Details'] as $product)
    {
      $this->products[ ] = new Product($product);
    }
    unset($soapclient);
    unset($soap_proxy);
  }
  else
  {
    // formata URL e chama parseXML para download e análise sintática
    $this->type = 'browse';
    $this->browseNode = $browseNode;
    $this->page = $page;
    $this->mode = $mode;
    $this->url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
               .'&dev-t='.DEVTAG.'&BrowseNodeSearch='
               .$this->browseNode.'&mode='.$this->mode
               .'&type=heavy&page='.$this->page
               .'&sort=+salesrank&f=xml';
    $this->parseXML( );
  }

  return $this->products;
}

// Dado um ASIN, obtém o URL da imagem grande
// Retorna uma string
function getImageUrlLarge($ASIN, $mode)
{
  foreach($this->products as $product)
  {
    if( $product->ASIN( )== $ASIN)
      return  $product->imageURLLarge( );
  }
  // se não for encontrado
```

Listagem 33.8 Continuação

```
  $this->ASINSearch($ASIN, $mode);
    return $this->products(0)->imageURLLarge( );
}

// Realiza uma consulta para obter um produto com ASIN especificado
// Alterna entre XML/HTTP e SOAP em constants.php
// Retorna um objeto Products
function ASINSearch($ASIN, $mode = 'books')
{
  $this->type = 'ASIN';
  $this->ASIN=$ASIN;
  $this->mode = $mode;
  $ASIN = padASIN($ASIN);

  if(METHOD=='SOAP')
  {
    error_reporting(E_ALL & ~E_NOTICE);
    $soapclient = new soapclient (
          'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
          'wsdl') ;
    $soap_proxy = $soapclient->getProxy( );
    $parameters['asin']=$ASIN;
    $parameters['mode']=$mode;
    $parameters['type']="heavy";
    $parameters['tag']=$this->assocID;
    $parameters['devtag']=$this->devTag;

    // realiza consulta soap real

    $result = $soap_proxy->AsinSearchRequest($parameters);
    if(isSOAPError($result))
    {
      print_r($result);
      return false;
    }
    $this->products[0] = new Product($result['Details'][0]);
    $this->totalResults=1;
    unset($soapclient);
    unset($soap_proxy);
  }
  else
  {
    // formata URL e chama parseXML para download e análise sintática
    $this->url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                 .'&dev-t='.DEVTAG.'&AsinSearch='
                 .$this->ASIN
                 .'&type=heavy&f=xml';
    $this->parseXML( );
  }
  return $this->products[0];
 }

// Realiza uma consulta para obter um página cheia de produtos com uma pesquisa
// Alterna entre XML/HTTP e SOAP em index.php
// Retorna um array de Products
function keywordSearch($search, $page, $mode = 'books')
{
  if(METHOD=='SOAP')
  {
    error_reporting(E_ALL & ~E_NOTICE);
```

Listagem 33.8 **Continuação**

```
      $soapclient = new soapclient(
           'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl','wsdl');
      $soap_proxy = $soapclient->getProxy( );
      $parameters['mode']=$mode;
      $parameters['page']=$page;
      $parameters['type']="heavy";
      $parameters['tag']=$this->assocID;
      $parameters['devtag']=$this->devTag;
      $parameters['sort']='+salesrank';
      $parameters['keyword'] = $search;

      // realiza a solicitação soap real
      $result = $soap_proxy->KeywordSearchRequest($parameters);

      if(isSOAPError($result) )
        return false;

      foreach($result['Details'] as $product)
      {
        $this->products[ ] = new Product($product);
      }
      $this->totalResults = $result['TotalResults'] ;
      unset($soapclient);
      unset($soap_proxy);
    }
    else
    {
      $this->type = 'search';
      $this->search=$search;
      $this->page = $page;
      $search = urlencode($search);
      $this->mode = $mode;
      $this->url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                   .'&dev-t='.DEVTAG.'&KeywordSearch='
                   .$search.'&mode='.$this->mode
                   .'&type=heavy&page='
                   .$this->page
                   .'&sort=+salesrank&f=xml';
      $this->parseXML( );
    }
    return $this->products;
  }

  // Analisa a XML no(s) objeto(s) Product
  function parseXML( )
  {
    // suprime erros porque isso às vezes falha
    $xml = @simplexml_load_file($this->url);
    if(!$xml)
    {
      //tenta uma segunda vez em caso de servidor ocupado
      $xml = @simplexml_load_file($this->url);
      if(!$xml)
      {
        return false;
      }

    }
    $this->totalResults = (integer)$xml->TotalResults;
    foreach($xml->Details as $productXML)
```

Listagem 33.8 Continuação

```
    {
      $this->products[ ] = new Product($productXML);
    }
  }
}
? >
```

Essa classe útil faz exatamente o tipo de coisa para o qual as classes são indicadas. Ela encapsula a interface para Amazon em uma bela caixa preta. Dentro da classe, a conexão com a Amazon pode ser feita via XML pelo método de HTTP ou de SOAP. O método que ela utilizará é determinado pela constante global `METHOD` que configuramos no início.

Vamos começar retornando ao exemplo Category Search. Utilizamos a classe `AmazonResultSet` como segue:

```
$ars = new AmazonResultSet;
$ars->browseNodeSearch($parameters['browsenode'],
                      $parameters['page'],
                      $parameters['mode']);
```

Essa classe não tem nenhum construtor, então iremos direto ao método `browseNodeSearch( )`. Estamos passando três parâmetros: o número do browsenode em que estamos interessados (correspondente a, digamos, Business & Investing ou Computers & Internet); o número da página, representando os registros que gostaríamos de ter recuperado; e o modo, representando o tipo de mercadoria em que estamos interessados. O código para esse método é mostrado em parte na Listagem 33.9.

Listagem 33.9 Método **browseNodeSearch( )** – Realizando uma pesquisa de categoria

```
function browseNodeSearch($browseNode, $page, $mode)
{
 if(METHOD=='SOAP')
 {
   $soapclient = new soapclient(
         'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
         'wsdl');
   $soap_proxy = $soapclient->getProxy( );
   $parameters['mode']=$mode;
   $parameters['page']=$page;
   $parameters['type']='heavy';
   $parameters['tag']=$this->assocID;
   $parameters['devtag']=$this->devTag;
   $parameters['sort']='+salesrank';
   $parameters['browse_node'] = $browseNode;

   // realiza a consulta soap real
   $result = $soap_proxy->BrowseNodeSearchRequest($parameters);
   if(isSOAPError($result))
     return false;
   $this->totalResults = $result['TotalResults'];

   foreach($result['Details'] as $product)
   {
     $this->products[ ] = new Product($product);
   }
   unset($soapclient);
   unset($soap_proxy);
 }
 else
```

Listagem 33.9 **Continuação**

```
  {
    // formata URL e chama parseXML para download e análise sintática
    $this->type = 'browse';
    $this->browseNode = $browseNode;
    $this->page = $page;
    $this->mode = $mode;
    $this->url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                .'&dev-t='.DEVTAG.'&BrowseNodeSearch='
                .$this->browseNode.'&mode='.$this->mode
                .'&type=heavy&page='.$this->page
                .'&sort=+salesrank&f=xml';
    $this->parseXML( );
  }

  return $this->products;
}
```

Dependendo do valor da constante METHOD, esse método executará a consulta via XML sobre HTTP ou via SOAP. Veremos cada um desses separadamente.

Utilizando REST/XML sobre HTTP

Para utilizar REST/XML sobre HTTP, comece configurando algumas variáveis de membro de classe importantes:

- type – O tipo de pesquisa exigido. Estamos procurando livros dentro de um browsenode particular; portanto, configuramos o valor como browse.
- browse – O valor do browsenode particular que passamos como um parâmetro.
- page – O número de página que passamos como um parâmetro.
- mode – O tipo de itens que estamos procurando (por exemplo, books) que passamos como um parâmetro.
- url – O URL na Amazon a que precisamos nos conectar a fim de realizar esse tipo de pesquisa.

Os URLs com que fizemos nossas conexões de HTTP para diferentes tipos de pesquisa e os parâmetros que eles esperam podem ser encontrados na Web Services API da Amazon.com e no Integration Guide em seu kit do desenvolvedor. Examine de perto nos parâmetros GET que estamos passando aqui:

```
$this->_url = 'http://xml.amazon.com/onca/xml2?t='.ASSOCIATEID
                .'&dev-t='.DEVTAG.'&BrowseNodeSearch='
                .$this->_browseNode.'&mode='.$this->_mode
                .'&type=heavy&page='.$this->_page
                .'&sort=+salesrank&f=xml';
```

Os parâmetros de que precisamos para passar para esse URL são:

- t – Seu Associate ID.
- dev-t – Seu token de desenvolvedor.
- BrowseNodeSearch – O número de browsenode que você deseja pesquisar.
- mode – books, ou outro tipo de produto válido.
- type – Heavy ou lite (note a ortografia!) Heavy fornece mais informações.
- page – Grupo de dez resultados.

- sort – A ordem em que gostaríamos que os resultados retornassem. Esse é um parâmetro opcional. Nesse caso, nós a configuramos como +salesrank porque gostaríamos que os resultados fossem organizados por ordem de classificação de vendas.
- f – O formato. Este deve sempre conter o valor 'xml'.

Tipos válidos são:

- Itens apresentados: +pmrank
- Mais vendidos: +salesrank
- Crítica média de cliente: +reviewrank
- Preço (do baixo para o alto): +pricerank
- Preço (do alto para o baixo): +inverse-pricerank
- Data de publicação: +daterank
- Ordem alfabética (A-Z): +titlerank
- Ordem alfabética (Z-A): -titlerank

Depois que todos esses parâmetros foram configurados, chamamos

```
$this->parseXML( );
```

para realmente fazer o trabalho. O método parseXML( ) é mostrado na Listagem 33.10.

Listagem 33.10 **Método parseXML( ) – Analisando sintaticamente a XML retornada de uma consulta**

```
// Analisa sintaticamente a XML para objeto(s) Product
function parseXML( )
{
 // suprime erros porque isso falha às vezes
 $xml = @simplexml_load_file($this->url);
 if(!$xml)
 {
   //tenta uma segunda vez em caso de o servidor estar ocupado
   $xml = @simplexml_load_file($this->url);
   if(!$xml)
   {
     return false;
   }

 }
 $this->totalResults = (integer)$xml->TotalResults;
 foreach($xml->Details as $productXML)
 {
   $this->products[ ] = new Product($productXML);
 }
}
```

A função simplexml_load_file( ) faz a maior parte do trabalho para você. Ela lê o conteúdo XML de um arquivo, ou, nesse caso, um URL. Ela fornece uma interface orientada a objetos com os dados e a estrutura no documento XML. Essa é uma interface útil para os dados, mas como você quer que um conjunto de funções de interface funcione para os dados que entram via REST ou SOAP, pode construir sua própria interface orientada a objetos para os dados nas instâncias da classe Product. Observe que você lança os atributos da XML para os tipos de variáveis PHP na versão

REST. Não se usa o operador cast no PHP, mas sem ele aqui, você receberia representações de objetos de cada dado inúteis para você.

A classe `Product` contém principalmente funções para acessar os dados armazenados em seus membros privados; portanto, não vale a pena imprimir o arquivo inteiro aqui, mas vale sim visitar a estrutura da classe e do construtor. A Listagem 33.11 contém parte da definição de `Product`.

Listagem 33.11 **A classe Product encapsula as informações que você tem sobre um produto da Amazon**

```
class Product
{
  private $ASIN;
  private $productName;
  private $releaseDate;
  private $manufacturer;
  private $imageUrlMedium;
  private $imageUrlLarge;
  private $listPrice;
  private $ourPrice;
  private $salesRank;
  private $availability;
  private $avgCustomerRating;
  private $authors = array( );
  private $reviews = array( );
  private $similarProducts = array( );

  function __construct($xml)
  {
    if(METHOD=='SOAP')
    {
      $this->ASIN = $xml['Asin'];
      $this->productName = $xml['ProductName'];
      if($xml['Authors'])
      {
        foreach($xml['Authors'] as $author)
        {
          $this->authors[ ] = $author;
        }
      }
      $this->releaseDate = $xml['ReleaseDate'];
      $this->manufacturer = $xml['Manufacturer'];
      $this->imageUrlMedium = $xml['ImageUrlMedium'];
      $this->imageUrlLarge = $xml['ImageUrlLarge'];

      $this->listPrice = $xml['ListPrice'];
      $this->listPrice = str_replace('$', '', $this->listPrice);
      $this->listPrice = str_replace(',', '', $this->listPrice);
      $this->listPrice = floatval($this->listPrice);

      $this->ourPrice = $xml['OurPrice'];
      $this->ourPrice = str_replace('$', '', $this->ourPrice);
      $this->ourPrice = str_replace(',', '', $this->ourPrice);
      $this->ourPrice = floatval($this->ourPrice);

      $this->salesRank = $xml['SalesRank'];
      $this->availability = $xml['Availability'];
      $this->avgCustomerRating = $xml['Reviews']['AvgCustomerRating'];
      $reviewCount = 0;
      if($xml['Reviews']['CustomerReviews'])
      {
```

Listagem 33.11 **Continuação**

```
        foreach ($xml['Reviews']['CustomerReviews'] as $review)
        {
          $this->reviews[$reviewCount]['rating'] = $review['Rating'];
          $this->reviews[$reviewCount]['summary'] = $review['Summary'];
          $this->reviews[$reviewCount]['comment'] = $review['Comment'];
          $reviewCount++;
        }
      }
      if($xml['SimilarProducts'])
      {
        foreach ($xml['SimilarProducts'] as $similar)
        {
          $this->similarProducts[ ] = $similar;
        }
      }
    }
    else // utilizando REST
    {
      $this->ASIN = (string)$xml->Asin;
      $this->productName = (string)$xml->ProductName;
      if($xml->Authors->Author)
      {
        foreach($xml->Authors->Author as $author)
        {
         $this->authors[ ] = (string)$author;
        }
      }
      $this->releaseDate = (string)$xml->ReleaseDate;
      $this->manufacturer = (string)$xml->Manufacturer;
      $this->imageUrlMedium = (string)$xml->ImageUrlMedium;
      $this->imageUrlLarge = (string)$xml->ImageUrlLarge;

      $this->listPrice = (string)$xml->ListPrice;
      $this->listPrice = str_replace('$', '', $this->listPrice);
      $this->listPrice = str_replace(',', '', $this->listPrice);
      $this->listPrice = floatval($this->listPrice);

      $this->ourPrice = (string)$xml->OurPrice;
      $this->ourPrice = str_replace('$', '', $this->ourPrice);
      $this->ourPrice = str_replace(',', '', $this->ourPrice);
      $this->ourPrice = floatval($this->ourPrice);

      $this->salesRank = (string)$xml->SalesRank;
      $this->availability = (string)$xml->Availability;
      $this->avgCustomerRating = (float)$xml->Reviews->AvgCustomerRating;
      $reviewCount = 0;
      if($xml->Reviews->CustomerReview)
      {
        foreach ($xml->Reviews->CustomerReview as $review)
        {
          $this->reviews[$reviewCount]['rating'] = (float)$review->Rating;
          $this->reviews[$reviewCount]['summary'] = (string)$review->Summary;
          $this->reviews[$reviewCount]['comment'] = (string)$review->Comment;
          $reviewCount++;
        }
      }
      if($xml->SimilarProducts->Product)
      {
        foreach ($xml->SimilarProducts->Product as $similar)
        {
```

Listagem 33.11 **Continuação**

```
            $this->similarProducts[ ] = (string)$similar;
          }
        }
      }
    }
```

Novamente, esse construtor pega duas formas diferentes de dados de entrada e cria uma interface de aplicação. Observe que embora uma parte do código poderia ficar mais genérica, alguns atributos mais complicados, como críticas (*reviews*), possuem diferentes nomes dependendo do método.

Após passar por todo esse processo para recuperar os dados, agora retornamos o controle de volta para a função `getARS( )` e daí de volta para `showBrowseNode( )`. O próximo passo é:

```
showSummary($ars->products( ), $page,
          $ars->totalResults( ), $mode,
          $browseNode);
```

A função `showSummary( )` simplesmente exibe os dados no `AmazonResultSet`, como vimos lá atrás na Figura 33.1. Portanto, não incluímos a função aqui.

Usando o SOAP

Vamos voltar e examinar a versão de SOAP da função `browseNodeSearch( )`. Repetiremos essa seção do código aqui:

```
$soapclient = new soapclient(
     'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
     'wsdl');
$soap_proxy = $soapclient->getProxy( );
$parameters['mode']=$mode;
$parameters['page']=$page;
$parameters['type']='heavy';
$parameters['tag']=$this->assocID;
$parameters['devtag']=$this->devTag;
$parameters['sort']='+salesrank';
$parameters['browse_node'] = $browseNode;

// realiza a consulta soap real
$result = $soap_proxy->BrowseNodeSearchRequest($parameters);
if(isSOAPError($result))
 return false;
$this->totalResults = $result['TotalResults'];

foreach($result['Details'] as $product)
{
 $this->products[ ] = new Product($product);
}
unset($soapclient);
unset($soap_proxy);
```

Não há nenhuma função extra pela qual passar aqui – o cliente de SOAP faz tudo para nós.

Começamos criando uma instância do cliente de SOAP:

```
$soapclient = new soapclient(
                'http://soap.amazon.com/schemas2/AmazonWebServices.wsdl',
                'wsdl');
```

Estamos fornecendo dois parâmetros ao cliente. O primeiro é a descrição WSDL do serviço e o segundo parâmetro é para dizer ao cliente de SOAP que isso é um URL em WSDL. Alternativa-

mente, acabamos de fornecer um parâmetro: o ponto final do serviço, que é o URL direto do servidor de SOAP.

Escolhemos fazer isso dessa maneira por uma boa razão, que está logo a seguir na próxima linha de código:

```
$soap_proxy = $soapclient->getProxy( );
```

Essa linha cria uma classe de acordo com as informações no documento de WSDL. Essa classe, o proxy de SOAP, terá métodos que correspondem aos métodos do serviço Web. Isso facilita muito a vida. Podemos interagir com o serviço Web como se fosse uma classe local do PHP.

Em seguida, configuramos um array dos parâmetros que precisamos passar para a consulta `browsenode`:

```
$parameters['mode']=$mode;
$parameters['page']=$page;
$parameters['type']='heavy';
$parameters['tag']=$this->_assocID;
$parameters['devtag']=$this->_devTag;
$parameters['sort']='+salesrank';
$parameters['browse_node'] = $browseNode;
```

Utilizando a classe `proxy`, podemos chamar os métodos de serviço Web passando o array de parâmetros:

```
$result = $soap_proxy->BrowseNodeSearchRequest($parameters);
```

Os dados armazenados em `$result` são um array que você pode armazenar diretamente como um objeto Product no array `products` na classe `AmazonResultSet`.

Gravando os dados em cache

Vamos voltar para a função `getARS( )` e falar sobre cache. Como você deve lembrar, a função tem a seguinte aparência:

```
// Obtém um AmazonResultSet do cache ou de um consulta ao vivo
// Se uma consulta ao vivo, acrescente-a ao cache
function getARS($type, $parameters)
{
  $cache = cached($type, $parameters);
  if($cache)  // se encontrado no cache
  {
    return  $cache;
  }
  else
  {
    $ars = new AmazonResultSet;
    if($type == 'asin')
      $ars->ASINSearch(padASIN($parameters['asin']), $parameters['mode']);
    if($type == 'browse')
      $ars->browseNodeSearch($parameters['browsenode'],
                             $parameters['page'], $parameters['mode']);
    if($type == 'search')
      $ars->keywordSearch($parameters['search'], $parameters['page'],
              $parameters['mode']);
    cache($type, $parameters, $ars);
  }
  return $ars;
}
```

Toda a gravação em cache do SOAP ou da XML da aplicação é feita por intermédio dessa função. Também temos outras imagens de cache. Começamos chamando a função cached( ) para ver se o AmazonResultSet exigido já está gravado em cache. Se estiver, retornamos esses dados em vez de fazer uma nova solicitação à Amazon:

```
$cache = cached($type, $parameters);
if($cache) // se encontrado no cache
{
  return $cache;
}
```

Se não estiver, quando obtivermos os dados de volta da Amazon, os adicionamos ao cache:

```
cache($type, $parameters, $ars);
```

Vejamos essas duas funções: cached( ) e cache( ). Elas são mostradas na Listagem 33.12. Essas funções implementam a gravação em cache que a Amazon exige como parte de seus termos e condições.

Listagem 33.12 **Funções cached( ) e cache( ) – Armazenando em cache funções de cachefunctions.php**

```
// Verifica se dados da Amazon estão em cache
// se estiverem, retorne-os
// se não, retorne false
function cached($type, $parameters)
{
  if($type == 'browse')
  {
    $filename =
➥CACHE.'/browse.'.$parameters['browsenode'].'.'.$parameters['page'].
              '.'.$parameters['mode'].'.dat';
  }
  if($type == 'search')
  {
    $filename = CACHE.'/search.'.$parameters['search'].'.'.$parameters['page'].
              '.'.$parameters['mode'].'.dat';
  }
  if($type == 'asin')
  {
    $filename = CACHE.'/asin.'.$parameters['asin'].'.'.$parameters['mode'].
              '.dat';
  }

  // os dados gravados em cache ausentes ou > 1 hora atrás?
  if(!file_exists($filename) ||
     ((mktime( ) - filemtime($filename)) > 60*60))
  {
    return false;
  }
  $data = file_get_contents($filename);
  return unserialize($data);
}

// adiciona dados da Amazon ao cache
function cache($type, $parameters, $data)
{
  if($type == 'browse')
  {
    $filename = CACHE.'/browse.'.$parameters['browsenode'].
              '.'.$parameters['page'].'.'.$parameters['mode'].'.dat';
```

Listagem 33.12 Continuação

```
  }
  if($type == 'search')
  {
    $filename = CACHE.'/search.'.$parameters['search'].'.'.$parameters['page'].
              '.'.$parameters['mode'].'.dat';
  }
  if($type == 'asin')
  {
    $filename = CACHE.'/asin.'.$parameters['asin'].'.'.$parameters['mode'].
              '.dat';
  }
  $data = serialize($data);

  $fp = fopen($filename, 'wb');
  if(!$fp||(fwrite($fp, $data)==-1))
  {
    echo ('<p>Error, could not store cache file');
  }
  fclose($fp);
}
```

Examinando esse código, você pode ver que esses arquivos de cache são armazenados sob um nome de arquivo que consiste no tipo da consulta seguido pelos parâmetros da consulta. A função `cache( )` armazena os resultados serializando-os, e a função `cached( )` os desserializa. A função `cached( )` também sobrescreverá quaisquer dados que tenham mais de uma hora, de acordo com os termos e condições.

A função `serialize( )` converte dados armazenados do programa em uma string que pode ser armazenada. Nesse caso, estamos criando uma representação armazenável de um objeto `AmazonResultSet`. Ao chamar `unserialize( )` ocorre o oposto, retornando a versão armazenada de volta para uma estrutura de dados na memória. Note que desserializar um objeto assim significa que você precisa ter a definição de classe no arquivo de modo que a classe seja compreensível e utilizável uma vez recarregada.

Em nossa aplicação, a recuperação de um conjunto de resultados do cache toma uma fração de um segundo. A criação de uma nova consulta ao vivo leva dez segundos.

Construindo o carrinho de compras

Então, dadas todas essas surpreendentes capacidades de consulta da Amazon, o que podemos fazer com elas? O mais óbvio é a possibilidade de criar um carrinho de compras. Como já abordamos esse tópico extensamente no Capítulo 27, não entraremos em detalhes aqui.

As funções do carrinho de compras são mostradas na Listagem 33.13.

Listagem 33.13 **cartfunctions.php** – Implementando o carrinho de compras

```
<?php
require_once('AmazonResultSet.php');

// Usando a função showSummary( ) na exibição do arquivo bookdisplay.php
// os conteúdos atuais do carrinho de compras
function showCart($cart, $mode)
{
  // constrói um array para passar
  $products = array( );
  foreach($cart as $ASIN=>$product)
  {
```

Listagem 33.13 **Continuação**

```
    $ars = getARS('asin', array('asin'=>$ASIN, 'mode'=>$mode));
    if($ars)
      $products[ ] = $ars->getProduct(0);
  }
  // constrõi o formulário para fazer o link um carrinho de compras da Amazon.com
  echo '<form method="POST"
            action="http://www.amazon.com/o/dt/assoc/handle-buy-box">';
  foreach($cart as $ASIN=>$product)
  {
    $quantity = $cart[$ASIN]['quantity'];
    echo "<input type='hidden' name='asin.$ASIN' value='$quantity'>";
  }
  echo '<input type="hidden" name="tag-value" value="ASSOCIATEID">';
  echo '<input type="hidden" name="tag_value" value="ASSOCIATEID">';
  echo '<input type="image" src="images/checkout.gif"
                        name="submit.add-to-cart"
                        value="Buy From Amazon.com">';
  echo ' When you have finished shopping press checkout to add all the
        items in your Tahuayo cart to your Amazon cart and complete
        your purchase.<br />';
  echo '</form>';

  echo '<a href = "index.php?action=emptycart"><img
          src = "images/emptycart.gif" alt = "Empty Cart" border = 0></a>
       If you have finished with this cart, you can empty it of all items.
       <br />';
  echo '<h1>Cart Contents</h1>';
  showSummary($products, 1, count($products), $mode,  0, true);

}

// mostra uma pequena visão geral do carrinho de compras que está sempre na tela
// mostra apenas os últimos três itens adicionados
function showSmallCart( )
{
  global $_SESSION;

  echo '<table border = 1 cellpadding = 1 cellspacing = 0>';
  echo '<tr><td class = cartheading>Your Cart $'.
      number_format(cartPrice( ), 2).
      '</td></tr>'<$I~Amazon;shopping carts;building><$I~shopping carts;building (Amazon)>;
  echo '<tr><td class = cart>'.cartContents( ).'</td></tr>';

  // formulário para criar link para um carrinho de compras da Amazon.com
  echo '<form method="POST"
            action="http://www.amazon.com/o/dt/assoc/handle-buy-box">';
  echo '<tr><td class = cartheading><a href =
                        "index.php?action=showcart"><img
                        src="images/details.gif" border=0></a>';
  foreach($_SESSION['cart'] as $ASIN=>$product)
  {
    $quantity = $_SESSION['cart'][$ASIN]['quantity'];
    echo "<input type='hidden' name='asin.$ASIN' value='$quantity'>";
  }
  echo '<input type="hidden" name="tag-value" value="ASSOCIATEID">';
  echo '<input type="hidden" name="tag_value" value="ASSOCIATEID">';
  echo '<input type="image" src="images/checkout.gif"
            name="submit.add-to-cart" value="Buy From Amazon.com">';
  echo '</td></tr>';
  echo '</form>';
```

Listagem 33.13 **Continuação**

```
  echo '</table>';
}

// mostra os três últimos itens adicionados ao carrinho
function cartContents( )
{
 global $_SESSION;

 $display = array_slice($_SESSION['cart'], -3, 3);
 // queremos na ordem cronológica inversa
 $display = array_reverse($display, true);

 $result = '';
 $counter = 0;

 // abrevia os nomes se forem longos
 foreach($display as $product)
 {
   if(strlen($product['name'])<=40)
     $result .= $product['name'].'<br />';
   else
     $result .= substr($product['name'], 0, 37).'...<br />';
   $counter++;
 }

 // adiciona linhas em branco se o carrinho estiver quase vazio para manter
 // a mesma exibição
 for(;$counter<3; $counter++)
 {
   $result .= '<br />';
 }
 return $result;
}

// calcula o preço total dos itens no carrinho
function cartPrice( )
{
 global $_SESSION;
 $total = 0.0;
 foreach($_SESSION['cart'] as $product)
 {
   $price = str_replace('$', '', $product['price']);
   $total += $price*$product['quantity'];
 }

 return $total;
}

// adiciona um único item ao carrinho
// atualmente não há facilidade para adicionar mais de um de cada vez
function addToCart(&$cart, $ASIN, $mode)
{
 if(isset($cart[$ASIN] ))
 {
   $cart[$ASIN]['quantity'] +=1;
 }
 else
 {
   // verifica se ASIN é válido e pesquisa o preço
   $ars = new AmazonResultSet;
```

Listagem 33.13 **Continuação**

```
    $product = $ars->ASINSearch($ASIN, $mode);

    if($product->valid( ))
      $cart[$ASIN] = array('price'=>$product->ourPrice( ),
                    'name' => $product->productName( ), 'quantity' => 1) ;
  }
}

// exclui todos de um item particular do carrinho
function deleteFromCart(&$cart, $ASIN)
{
  unset ($cart[$ASIN]);
}
?>
```

Existem algumas diferenças em relação à maneira como fazemos as coisas com esse carrinho. Por exemplo, veja a função `addToCart( )`. Quando tentamos adicionar um item ao carrinho, podemos verificar se ele tem um ASIN válido e pesquisar o preço atual (ou pelo menos, gravado em cache).

O que é realmente interessante aqui é essa pergunta: quando os clientes concluem a compra, como obtemos seus dados para a Amazon?

Verificando a Amazon

Olhe atentamente a função `showCart( )` na Listagem 33.13. Aqui está a parte relevante:

```
// constrói o formulário para fazer o link um carrinho de compras da Amazon.com
echo '<form method="POST"
          action="http://www.amazon.com/o/dt/assoc/handle-buy-box">';
foreach($cart as $ASIN=>$product)
{
  $quantity = $cart[$ASIN]['quantity'];
  echo "<input type='hidden' name='asin.$ASIN' value='$quantity'>";
}
echo '<input type="hidden" name="tag-value" value="ASSOCIATEID">';
echo '<input type="hidden" name="tag_value" value="ASSOCIATEID">';
echo '<input type="image" src="images/checkout.gif"
                         name="submit.add-to-cart"
                         value="Buy From Amazon.com">';
echo ' When you have finished shopping press checkout to add all the
       items in your Tahuayo cart to your Amazon cart and complete
       your purchase.<br />';
echo '</form>';
```

O botão checkout é um botão do formulário que conecta o carrinho do cliente ao carrinho de compras da Amazon. Enviamos ASINs, quantidades e nosso Associate ID por meio de variáveis `POST`. E pronto! Você pode ver o resultado final de clicar nesse botão na Figura 33.5, no início deste capítulo.

Uma dificuldade com essa interface é que ela é uma interação de uma via. Podemos adicionar itens ao carrinho da Amazon, mas não podemos removê-los. Isso significa que as pessoas não podem navegar facilmente de um lado a outro entre os sites sem acabar com itens duplicados em seus carrinhos.

Instalando o código do projeto

Se quiser instalar o código do projeto deste capítulo, você precisará dar alguns passos além do normal. Depois que tiver o código em um local apropriado em seu servidor, você precisará fazer o seguinte:

- Crie um diretório de cache.
- Configure as permissões no diretório de cache de modo que os scripts sejam capazes de gravar nele.
- Edite `constants.php` para fornecer a localização do cache.
- Inscreva-se para obter uma tag de desenvolvedor da Amazon.
- Edite `constants.php` para incluir sua tag de desenvolvedor e, opcionalmente, seu Associate ID.
- Certifique-se de que NuSOAP esteja instalado. Está dentro do diretório Tahuayo, mas você pode movê-lo para alterar o código.
- Verifique se você compilou o PHP com suporte de XML.

Estendendo o projeto

Há uma muita coisa divertida que você poderia fazer para estender esse projeto:

- Poderia expandir os tipos de pesquisas que estão disponíveis via Tahuayo.
- Talvez você gostaria de experimentar o XSLT Web Service da Amazon.
- Poderia verificar os links para amostras de aplicações inovadoras no Web Services How-To da Amazon. Dê uma olhada nestas aplicações para ter mais idéias:

http://associates.amazon.com/exec/panama/associates/ntg/browse/-/567634/
Os carrinhos de compras são a construção mais óbvia com esses dados, mas não são a única.

Leitura adicional

Há um milhão de livros e recursos on-line disponíveis sobre os tópicos XML e serviços Web. Um ótimo lugar para começar é sempre o W3C. Você também pode visitar a página XML Working Group:

http://www.w3.org/XML/Core/

e a página Web Services Activity:

http://www.w3.org/2002/ws/

apenas como uma introdução.

VI

Apêndices

A

Instalando o PHP e o MySQL

APACHE, PHP E MySQL estão disponíveis para diferentes combinações de sistemas operacionais e servidores Web. Neste apêndice, explicaremos como configurar o Apache, o PHP e o MySQL em algumas plataformas de servidor. Abordaremos as opções mais comuns disponíveis para Unix e Windows XP.

Os tópicos-chave que abordaremos neste apêndice incluem:

- Executar o PHP como um interpretador de CGI ou como um módulo;
- Instalar o Apache, o SSL, o PHP e o MySQL sob o Unix;
- Instalar o Apache, o PHP e o MySQL sob o Windows;
- Testar se está funcionando usando `phpinfo( );`
- Adicionar o PHP e o MySQL ao Microsoft Internet Information Server;
- Instalar PEAR;
- Considerar outras configurações.

Nosso objetivo neste apêndice é fornecer um guia de instalação para um servidor Web que permitirá hospedar diversos Web sites. Alguns sites, como os exemplos abrangidos, requerem Secure Sockets Layer (SSL) para soluções de comércio eletrônico. E a maioria é baseada via scripts para conectar-se a um servidor de banco de dados (DB) e extrair e processar dados.

Muitos usuários do PHP nunca precisam instalar o PHP em uma máquina, razão pela qual esse material está em um apêndice em vez de no Capítulo 1. A maneira mais fácil de obter acesso a um servidor confiável com uma conexão de Internet rápida e com o PHP já instalado é simplesmente inscrever-se em uma conta em um dos milhares de serviços de hospedagem ou revendedores de serviço de hospedagem espalhados pelo globo.

Dependendo da razão por que você está instalando o PHP em uma máquina, talvez tome decisões diferentes. Se você tiver uma máquina permanentemente conectada à rede que pretende utilizar como um servidor ao vivo, o desempenho será importante. Se você estiver construindo um servidor de desenvolvimento para criar e testar código, ter uma configuração semelhante ao do servidor ao vivo será o ponto mais importante. Se pretende executar ASP e PHP na mesma máquina, limitações diferentes serão aplicadas.

Executando PHP como um interpretador ou módulo de CGI

O interpretador do PHP pode ser executado como um módulo ou como um binário CGI separado. Em geral, a versão de módulo é utilizada por razões de desempenho. Mas a versão CGI é eventual-

mente utilizada para servidores para os quais uma versão do módulo não está disponível ou porque permite que os usuários do Apache executem diferentes páginas capacitadas para trabalhar com o PHP com diferentes IDs de usuário.

Neste apêndice, abordaremos principalmente a opção de módulo como o método para executar o PHP.

Instalando Apache, PHP e MySQL sob Unix

Dependendo das suas necessidades e do seu nível de experiência com sistemas Unix, talvez você opte por uma instalação binária ou por compilar os programas diretamente a partir dos seus códigos-fonte. As duas abordagens têm vantagens.

Uma instalação binária demandará alguns minutos para um especialista, e para um iniciante, um pouco mais de tempo, mas resultará em um sistema que provavelmente está uma versão ou duas atrás das distribuições atuais e em um sistema configurado com as opções de uma outra pessoa.

Uma instalação baseada no código-fonte exigirá algumas horas para o download, instalação e configuração e, nas primeiras vezes, causa uma certa apreensão. Mas isso proporciona a você um controle completo. Você escolhe o que instalar, quais versões utilizar e quais diretivas de configuração especificar.

Instalação binária

A maioria das distribuições do Linux inclui um servidor da Web Apache pré-configurado com o PHP predefinido. Os detalhes sobre o que é fornecido dependerão da sua opção de distribuição e de versão.

Uma desvantagem das instalações binárias é que você raramente obtém a última versão do programa. Dependendo da importância das últimas distribuições de correção de bugs, obter uma versão antiga pode não ser um problema para você. O maior problema é que você não pode escolher que opções são compiladas nos seus programas.

O caminho mais confiável e flexível a tomar é compilar todos os programas que você precisa a partir do código-fonte deles. Isso levará um pouco mais tempo do que instalar os RPMs; portanto, você talvez opte por utilizar os RPMs ou outros pacotes binários se disponíveis. Mesmo se os arquivos binários não estiverem disponíveis a partir de fontes oficiais com a configuração de que você precisa, talvez você seja capaz de encontrar fontes não-oficiais com um mecanismo de pesquisa.

Instalação do código-fonte

Agora, vamos instalar o Apache, o PHP e o MySQL em um ambiente Unix. Primeiro, precisamos decidir quais módulos extras carregaremos sob o trio. Como alguns exemplos abrangidos neste livro exigem utilizar um servidor seguro para transações pela Web, instalaremos um servidor capacitado para trabalhar com SSL (Secure Sockets Layer).

Nossa configuração do PHP será mais ou menos a configuração padrão, mas também abrangerá a ativação das duas seguintes bibliotecas sob o PHP.

- gd2
- PDFlib

Essas são somente duas das muitas bibliotecas disponíveis para o PHP. Então, estamos ativando-as para que você possa ter uma idéia do que é necessário para ativar bibliotecas extras dentro do PHP. A compilação da maioria dos programas Unix segue um processo semelhante.

Normalmente, é necessário recompilar o PHP depois de instalar uma nova biblioteca; assim, se você souber do que precisa antecipadamente, instale todas as bibliotecas requeridas na máquina e então comece a compilar o módulo do PHP.

Aqui, descrevemos a instalação em um servidor SuSE Linux, mas a descrição é genérica e pode ser aplicada em outros servidores Unix.

Vamos começar reunindo os arquivos requeridos para nossa instalação. Precisamos desses programas:

- Apache (*http://www.apache.org/*): O servidor Web.
- OpenSSL (*http://www.openssl.org/*): O conjunto de ferramentas de código-fonte aberto que implementa a Secure Sockets Layer.
- Mod_SSL (*http://www.modssl.org/*): Fornece uma interface do módulo Apache para OpenSSL.
- MySQL (*http://www.mysql.com/*): O banco de dados relacional.
- PHP (*http://www.php.net/*): A linguagem de criação de scripts do lado do servidor.
- *http://www.pdflib.com/products/pdflib/download/index.html*: Biblioteca para gerar documentos PDF instantaneamente.
- *ftp://ftp.uu.net/graphics/jpeg/*: A biblioteca JPEG, necessária para PDFlib e gd.
- *http://www.libpng.org/pub/png/libpng.html*: A biblioteca PNG, necessária para gd.
- *http://www.gzip.org/zlib/*: A biblioteca zlib, necessária para a biblioteca PNG, descrita anteriormente.
- *http://www.libtiff.org/*: A biblioteca TIFF, necessária para PDFlib.
- *ftp://ftp.cac.washington.edu/imap/*: O cliente c IMAP, necessário para IMAP.

Se você quiser utilizar a função `mail( )`, precisará ter um MTA (mail transfer agent) instalado, embora não tratemos desse assunto aqui.

Assumiremos que você tem acesso de root ao servidor e as seguintes ferramentas instaladas em seu sistema:

- `gzip` ou `gunzip`
- `gcc` e GNU `make`

Quando estiver pronto para começar o processo de instalação, você deve iniciar fazendo download de todas as fontes de arquivo `tar` para um diretório temporário. Certifique-se de os colocar em algum lugar com bastante espaço. Em nosso caso, escolhemos `/usr/srcpara`, o diretório temporário. Você deve fazer download deles como root para evitar problemas de permissões.

Instalando o MySQL

Nesta seção, mostraremos como realizar uma instalação binária do MySQL. Isso automaticamente colocará os arquivos em vários lugares. Os diretórios que escolhemos para o resto do nosso trio são:

- `/usr/local/apache`
- `/usr/local/ssl`

Você pode instalar as aplicações em diretórios diferentes alterando a opção prefix antes da instalação.

Vamos começar! Torne-se root utilizando `su`

```
$ su root
```

e insira a senha do usuário root. Mude para o diretório em que você armazenou os arquivos-fonte, por exemplo:

```
# cd /usr/src
```

O MySQL recomenda que você faça download de um binário do MySQL em vez de compilar a partir do zero. Que versão utilizar depende do que você deseja fazer. No momento em que escrevíamos este livro, a versão de produção do MySQL era a 4.0. Você precisa da 4.1 se quiser utilizar subconsultas, e da 5.0 se quiser utilizar procedures armazenadas. Quando você for ler este livro, uma dessas versões pode ter se tornado a versão de produção. Embora as versões pré-lançamento do MySQL sejam geralmente bastante estáveis, você pode escolher não utilizá-las em um site de produção. Se estiver aprendendo e experimentando na sua própria máquina, pode escolher utilizar uma dessas versões.

Faça o download dos seguintes pacotes:

```
MySQL-server-VERSION.i386.rpm
MySQL-Max-VERSION.i386.rpm
MySQL-client-VERSION.i386.rpm
```

(A palavra *VERSION* deve ser substuída pelo número da versão. Seja qual for a versão escolhida, certifique-se de escolher um conjunto correspondente.) Se pretende executar o cliente e servidor MySQL nessa máquina e compiliar o suporte MySQL para outros programas, como o PHP, você precisará de todos esses pacotes.

Digite os seguintes comandos para instalar os servidores e cliente MySQL:

```
rpm -i MySQL-server-VERSION.i386.rpm
rpm -i MySQL-Max-VERSION.i386.rpm
rpm -I MySQL-client-VERSION.i386.rpm
```

O servidor MySQL deve estar executando sem problemas.

Agora é hora de fornecer ao usuário root uma senha. Substitua `new-password` no seguinte comando com uma senha de sua escolha; caso contrário, `new-password` será a senha do root:

```
mysqladmin -u root password 'new-password'
```

Quando você instalar o MySQL, ele automaticamente criará dois bancos de dados. Um é a tabela `mysql`, que controla usuários, hosts e permissões DB no servidor real. O outro é um DB de teste. Você pode marcar seu banco de dados via linha de comando assim:

```
# mysql -u root -p
Enter password:
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| mysql              |
| test               |
+--------------------+
2 rows in set (0.00 sec)
```

Digite **quit** ou **\q** para fechar o cliente de MySQL.

A configuração padrão do MySQL permite que qualquer usuário acesse o sistema sem fornecer um nome de usuário ou senha. Isso obviamente é indesejável.

A parte final obrigatória da faxina do MySQL é excluir o usuário anônimo. Abrir um prompt de comando e digitar as seguintes linhas realizará isso.

```
# mysql -u root -p
mysql> use mysql
mysql> delete from user where User='';
mysql> quit
```

Em seguida, você precisará digitar

```
mysqladmin -u root -p reload
```

para que essas alterações tenham efeito.

Você também deve ativar log binário no servidor MySQL porque precisará dele se planeja utilizar replicação. Para isso, primeiro pare o servidor:

```
mysqladmin -u root -p shutdown
```

Crie um arquivo chamado `/etc/my.cnf` para ser usado como seu arquivo de opções MySQL. Nesse momento, você precisa apenas de uma opção, mas pode definir várias aqui. Consulte o manual do MySQL para obter uma lista completa.

Abra o arquivo e digite:

```
[mysqld]
log-bin
```

Salve o arquivo e saia. Então, reinicie o servidor executando `mysqld_safe`.

Instalando o PDFlib

Se quiser utilizar PDFlib para criar arquivos PDF, como discutimos no Capítulo 32, você pode pular esta seção.

Faça download da PDFlib em http://www.pdflib.com/products/pdflib/download/index.html.

Para extrair o conteúdo do repositório de arquivo PDFlib, digite:

```
# gunzip -c PDFlib-6.0.0p1-Linux.tar.gz | tar xvf -
```

Não utilizamos a PDFlib diretamente; portanto, voltaremos a ela depois que o PHP estiver executando.

Instalando o PHP

Você deve ainda estar atuando como root, se não su volta para root.

Antes que possamos instalar o PHP, você precisa ter o Apache pré-configurado de modo que saiba o local em que tudo está. Você voltará a isso mais adiante na seção quando configurar o servidor Apache. Mude outra vez para o diretório em que estão os códigos-fonte.

```
# cd /usr/src
# gunzip -c apache_1.3.31.tar.gz | tar xvf -
# cd apache_1.3.31
# ./configure --prefix=/usr/local/apache
```

Certo, agora você pode começar a configurar o PHP. Extraia os arquivos-fonte e mude para seu diretório:

```
# cd /usr/src
# gunzip -c php-5.0.0.tar.gz | tar xvf -
#  cd php-5.0.0
```

Novamente, há muitas opções com o comando `configure` do PHP. Utilize `./configure --help=short` para determinar o que deseja adicionar. Nesse caso, queremos adicionar suporte ao MySQL, PDFLib, Apache e gd.

Observe que tudo a seguir é um comando. Podemos colocá-lo todo em uma linha ou, como temos aqui, utilizar o caractere de continuação, barra invertida (\), para permitir digitar um comando com diversas linhas para melhorar a legibilidade.

```
# ./configure --with-mysqli=mysql_config_path/mysql_config \
            --with-apache=../apache_1.3.31 \
            --with-jpeg-dir=/path/to/jpeglib \
            --with-tiff-dir=/path/to.tiffdir \
            --with-zlib-dir=/path/to/zlib \
            --with-imap=/path/to/imapcclient \
            --with-gd
```

Em seguida, faça e instale os binários:

```
# make
# make install
```

Copie um arquivo INI para o diretório `lib`

```
#  cp php.ini-dist /usr/local/lib/php.ini
```

ou

```
# cp php.ini-recommended /usr/local/lib/php.ini
```

As duas versões do `php.ini` nos comandos sugeridos têm conjuntos diferentes de opções. A primeira, `php.ini-dist`, é concebida para máquinas de desenvolvimento. Por exemplo, ela tem `display_errors` configurado como On. Isso tornará o desenvolvimento mais fácil, mas não será realmente apropriado em uma máquina de produção. Quando nos referimos a um valor padrão da configuração do `php.ini` neste livro, tínhamos em mente sua configuração nessa versão do `php.ini`. A segunda versão, `php.ini-recommended`, destina-se a máquinas de produção.

Você pode editar o arquivo `php.ini` para configurar as opções do PHP. Há várias opções que você talvez opte por configurar, mas algumas em particular merecem menção. Você talvez precise configurar o valor de `sendmail_path` se quiser enviar e-mail a partir de scripts.

É hora de configurar OpenSSL. Isso é o que você utilizará para criar certificados temporários e arquivos CSR. O `--prefix` especifica o diretório principal de instalação.

```
# gunzip -c openssl-0.9.7d.tar.gz | tar xvf -
# cd openssl-0.9.7d
# ./config --prefix=/usr/local/ssl
```

Agora execute, teste e instale:

```
# make
# make test
# make install
```

A seguir, configuraremos o módulo mod_SSL e então o especificaremos para ser um módulo carregável com a configuração do Apache.

```
# cd /usr/src/
# gunzip -c mod_ssl-2.8.11-1.3.31.tar.gz |tar xvf -
# cd mod_ssl-2.8.18-1.3.31
# ./configure --with-apache=../apache_1.3.31
```

Observe que podemos adicionar mais módulos Apache à árvore de fontes do Apache. A opção `--enable-shared=ssl` opcional ativa a construção de mod_SSL como um DSO `libssl.so`. Leia os documentos `INSTALL` e `htdocs/manual/dso.html` na árvore de origem do Apache para informações adicionais sobre suporte de DSO no Apache. É bastante recomendável que os ISPs e mantenedores de pacotes utilizem o recurso DSO para flexibilidade máxima com mod_SSL. Note, porém, que o Apache não suporta DSO em todas as plataformas.

```
# cd ../apache_1.3.31
#  SSL_BASE=../openssl-0.9.7d \
    ./configure \
    --enable-module=ssl \
    --activate-module=src/modules/php5/libphp5.a \
    --prefix=/usr/local/apache \
     --enable-shared=ssl
```

Por fim você pode criar os arquivos do Apache e os certificados e então instalá-los:

```
# make
```

Se você fez tudo certo, receberá uma mensagem semelhante à seguinte:

```
+---------------------------------------------------------------------+
| Before you install the package you now should prepare the SSL       |
| certificate system by running the 'make certificate' command.       |
| For different situations the following variants are provided:      |
|                                                                     |
| % make certificate TYPE=dummy    (dummy self-signed Snake Oil cert) |
| % make certificate TYPE=test     (test cert signed by Snake Oil CA) |
| % make certificate TYPE=custom   (custom cert signed by own CA)     |
| % make certificate TYPE=existing (existing cert)                    |
|        CRT=/path/to/your.crt [KEY=/path/to/your.key]                |
|                                                                     |
| Use TYPE=dummy    when you're a  vendor package maintainer,         |
| the TYPE=test     when you're an admin but want to do tests only,   |
| the TYPE=custom   when you're an admin willing to run a real server |
| and TYPE=existing when you're an admin who upgrades a server.       |
| (The default is TYPE=test)                                          |
|                                                                     |
| Additionally add ALGO=RSA (default) or ALGO=DSA to select           |
| the signature algorithm used for the generated certificate.         |
|                                                                     |
| Use 'make certificate VIEW=1' to display the generated data.        |
|                                                                     |
| Thanks for using Apache & mod_ssl.       Ralf S. Engelschall        |
|                                          rse@engelschall.com        |
|                                          www.engelschall.com        |
+---------------------------------------------------------------------+
```

Agora você pode criar um certificado personalizado. Essa opção apresentará um prompt pedindo a localização, a empresa e outros poucos detalhes. Para informações de contato, faz sentido utilizar dados reais. Para outras perguntas durante o processo, a resposta padrão é suficiente.

```
# make certificate TYPE=custom
```

Agora instale o Apache:

```
# make install
```

Se tudo correu bem, a mensagem que você deve ver é parecida a esta:

```
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|   /usr/local/apache/conf/httpd.conf                    |
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|   /usr/local/apache/bin/apachectl start                |
|                                                        |
| Or when you want to run it with SSL enabled use:       |
|                                                        |
|   /usr/local/apache/bin/apachectl startssl             |
|                                                        |
| Thanks for using Apache.       The Apache Group        |
|                                http://www.apache.org/  |
+--------------------------------------------------------+
```

Agora está na hora de ver se o Apache e o PHP estão funcionando. Entretanto, precisamos editar o `httpd.conf` para adicionar o tipo do PHP à configuração.

Arquivo **httpd.conf** – Fragmentos

Examine o `httpd.conf` e adicione ou remova o caractere de comentário das seguintes linhas. Se você seguiu as instruções anteriores, seu arquivo `httpd.conf` estará localizado no diretório `/usr/local/apache/conf`. O arquivo tem o `addtype` para o PHP comentado. Você deve remover o caractere de comentário dele neste momento, de modo que pareça isto:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Agora estamos prontos para iniciar o servidor Apache para ver se ele funcionou. Primeiro, iniciaremos o servidor sem o suporte de SSL para ver se ele avança. Verificaremos suporte de PHP e então pararemos o servidor e iniciaremos com o suporte SSL ativado e veremos se conseguimos fazer tudo funcionar.

`configtest` verificará se toda a instalação está corretamente configurada:

```
# cd /usr/local/apache/bin
# ./apachectl configtest
Syntax OK
# ./apachectl start
   ./apachectl start: httpd started
```

Se tudo funcionar corretamente, você verá algo semelhante à Figura A.1 ao se conectar ao servidor com um navegador Web.

Nota Você conecta-se ao servidor com um nome de domínio ou utilizando o endereço de IP real do computador. Verifique ambos os casos, para assegurar que tudo esteja funcionando adequadamente.

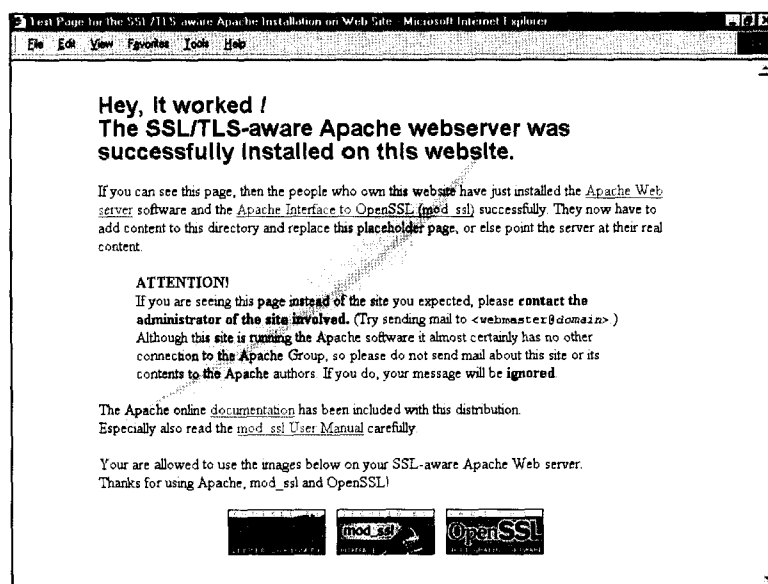



Figura A.1 A página de teste padrão fornecida pelo Apache.

O suporte de PHP está funcionando?

Agora testaremos o suporte do PHP. Crie um arquivo com o nome de `test.php` com o seguinte código. O arquivo precisa estar localizado no caminho da raiz de documentos, que deve estar configura-

do por padrão como /usr/local/apache/htdocs. Observe que isso é dependente do prefixo de diretório que escolhemos inicialmente. Entretanto, isso poderia ser alterado no httpd.conf.

```
<?php phpinfo( ); ?>
```

A tela de saída deve ser parecida com a Figura A.2.

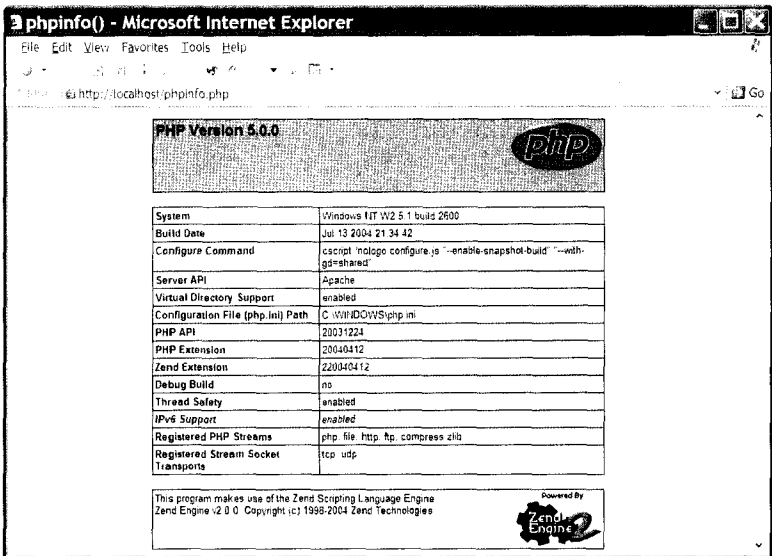



Figura A.2 A função phpinfo( ) fornece informações úteis de configuração.

O SSL está funcionando?

Certo, agora estamos prontos para testar o SSL. Primeiro, pare o servidor e reinicie com a opção SSL ativada:

```
# /usr/local/apache/bin/apachectl stop
# /usr/local/apache/bin/apachectl startssl
```

Teste para ver se ele funciona, conectando-se ao servidor com um navegador Web e selecionando o protocolo https, assim:

https://*seuservidor.seudomínio*.com

Tente também o endereço de IP do seu servidor, assim:

https://xxx.xxx.xxx.xxx

ou

http://xxx.xxx.xxx.xxx:443

Se funcionou, o servidor enviará o certificado para o navegador estabelecer uma conexão segura. Isso fará o navegador apresentar um prompt pedindo para você aceitar o certificado auto-assinado. Se esse fosse um certificado de uma autoridade de certificação em que seu navegador já confiasse, ele não o alertaria. Em nosso caso, criamos e assinamos nossos próprios certificados. Não quisemos comprar um imediatamente. Primeiro, quisemos nos assegurar de que poderíamos fazer tudo funcionar adequadamente.

Se estiver utilizando Internet Explorer ou Netscape, você verá o símbolo de um cadeado na barra de status. Isso informa que foi estabelecida uma conexão segura. O ícone utilizado pelo Netscape é mostrado na Figura A.3.

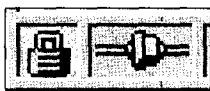



Figura A.3 Os navegadores Web exibem um ícone para indicar que a página que você está visualizando veio via uma conexão de SSL.

Passos finais

A fim de utilizar o objeto compartilhado PDFlib e quaisquer outros módulos que instalou dessa maneira, você precisará completar alguns outros passos.

Copie o arquivo `libpdf_php` para o diretório de extensões do PHP. Isso provavelmente será:

```
/usr/local/lib/php/extensions
```

Adicione a seguinte linha para seu arquivo do `php.ini`:

```
extension = libpdf_php.so
```

Instalando o Apache, o PHP e o MySQL sob Windows

Com o Windows, o processo de instalação é um pouquinho diferente porque o PHP é configurado como um script CGI (`php.exe`) ou como um módulo ISAPI (`php5isapi.dll`). Mas o Apache e o MySQL são instalados de maneira semelhante à maneira como são instalados sob o Unix. Certifique-se de ter os últimos patches do serviço do sistema operacional aplicados à máquina antes de iniciar a instalação do Windows.

Você deve iniciar fazendo download de todos os últimos arquivos-fonte para um diretório temporário com amplo espaço. Para nossa instalação, utilizamos `c:\temp\download` como nosso diretório temporário.

Se tiver uma conexão de rede lenta, talvez você prefira utilizar as versões do CD, mas possivelmente haverá uma ou mais versões obsoletas.

Instalando MySQL sob Windows

As seguintes instruções foram escritas utilizando Windows XP.

Comece configurando o MySQL. Você pode fazer download do arquivo ZIP necessário em http://www.mysql.com.

Descompacte o arquivo ZIP MySQL para o arquivo temporário e execute o programa `Setup.exe`. O instalador é um InstallShield Wizard padrão e deve se parecer com muitos outros instaladores que você conhece.

Escolher Typical Install no assistente não fará nenhuma pergunta, além do local onde você gostaria de instalar o MySQL. O diretório em que o MySQL é instalado por padrão será o diretório `C:\mysql`. Você pode movê-lo para um diretório diferente se necessário, depois que ele estiver completamente instalado, mas se fizer isso, você precisará seguir alguns passos extras para manter tudo em funcionamento.

Se move o MySQL e pretende executar o executável do MySQL, o `mysqld`, você deverá informar o local onde tudo estará, fornecendo as opções da linha de comando. Utilize `C:\mysql\bin\mysqld --help` para exibir todas as opções.

Por exemplo, se tiver movido a distribuição do MySQL para `D:\programs\mysql`, você deve iniciar `mysqld` com:

```
D:\programs\mysql\bin\mysqld --basedir D:\programs\mysql
```

Se mover a instalação e executá-la como um serviço do Windows, você precisará criar um arquivo INI chamado `my.ini` e colocá-lo no seu diretório principal do Windows. Seu arquivo INI terá algum conteúdo parecido com isto:

```
[mysqld]
basedir=D:/programs/mysql/bin/
datadir= D:/programs/mysql/data/
```

Na configuração NT/2000/XP, o nome do servidor MySQL é `mysqld-nt` e ele normalmente será instalado como um serviço. Um serviço é um programa que é executado constantemente no segundo plano e se destina a fornecer serviços a outros programas. Normalmente, eles são executados automaticamente quando você inicia a máquina, o que lhe poupará o trabalho de iniciá-los todas as vezes.

Você pode instalar o servidor MySQL como um serviço indo ao prompt de comando do Windows e digitando isto:

```
cd c:\mysql\bin
mysqld-nt --install
```

A resposta que você deve obter é:

```
Service successfully installed.
```

Agora você pode iniciar e parar o serviço MySQL a partir da linha de comando com:

```
NET START mysql
NET STOP mysql
```

Observe que o nome do executável é `mysqld-nt`, mas o nome do serviço é apenas `mySQL`. Se executar `NET START mySQL`, você deverá ver a seguinte mensagem:

```
The MySql service was started successfully.
```

Depois que o servidor foi instalado, ele pode ser parado, iniciado ou configurado para iniciar automaticamente utilizando o utilitário Services (localizado no Control Panel). Para abrir Services, clique em Start, aponte para Settings e clique no Control Panel. Dê um clique duplo em Administrative Tools e, então, em Services.

O utilitário Services é mostrado na Figura A.4. Se quiser configurar quaisquer opções do MySQL, você deve primeiro parar o serviço e então especificá-las como *startup parameters* no utilitário Services antes de reiniciar o serviço MySQL. O serviço MySQL pode ser parado utilizando o utilitário Services ou com os comandos `NET STOP MySQL` ou `mysqladmin shutdown`.

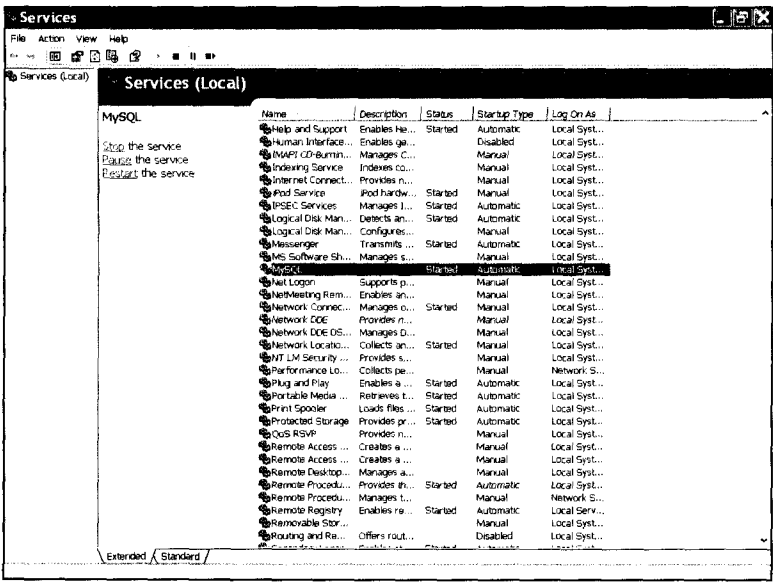



Figura A.4 O utilitário Services permite configurar os serviços que executam na sua máquina.

Para testar se o MySQL está funcionando, pode-se executar o seguinte comando:

```
C:\mysql\bin\mysqlshow
```

A configuração padrão não é realmente a ideal. Há alguns problemas que precisamos considerar:

- Configurar o `PATH`
- Excluir o usuário anônimo
- Configurar a senha root

Configurando o PATH

O MySQL vem com uma grande quantidade de utilitários de linha de comando com diferentes graus de utilidade. Nenhum deles é fácil de conseguir, a menos que o diretório binário do MySQL esteja no seu `PATH`. O propósito dessa variável de ambiente é informar ao Windows o local onde procurar programas executáveis.

Muitos dos comandos comuns que você utiliza no prompt de comando do Windows, como `dir` e `cd`, são internos e incorporados ao `cmd.exe`. Outros, como `format` e `ipconfig`, têm seus próprios executáveis. Não seria conveniente ter de digitar **`C:\WINNT\system32\f`** se você quisesse formatar um disco; nem ter de digitar **`C:\mysql\bin\mysql`** para executar o monitor do MySQL.

O diretório em que os executáveis para seus comandos básicos do Windows, como `format.exe`, residem está automaticamente no seu `PATH`; portanto, você simplesmente pode digitar **`format`**. A fim de ter a mesma conveniência com as ferramentas da linha de comando `mysql`, precisamos adicioná-la.

Clique em Start e escolha Settings, Control Panel. Dê um clique duplo em System e vá à guia Advanced. Se você clicar no botão Environment Variables, será exibida uma caixa de diálogo que permite visualizar as variáveis de ambiente do seu sistema. Dar um clique duplo em `PATH` permite editá-lo.

Adicione um ponto-e-vírgula ao final do seu caminho atual para separar sua nova entrada da anterior, em seguida adicione `c:\mysql\bin`. Quando você clica em OK, sua adição será armazenada no registro da máquina. Na próxima vez em que reiniciar a máquina, você será capaz de digitar **`mysql`** em vez de **`C:\mysql\bin\mysql`**.

Excluindo o usuário anônimo

A configuração padrão do MySQL permite que qualquer usuário acesse o sistema sem fornecer um nome de usuário ou senha. Isso obviamente é indesejável.

A primeira coisa que queremos fazer é excluir o usuário anônimo. Abrir um prompt de comando e digitar as seguintes linhas realizará isso:

```
c:\mysql\bin\mysql
use mysql
delete from user where User='';
quit
c:\mysql\bin\mysqladmin reload
```

Agora, o usuário anônimo desapareceu.

Configurando a senha de root

Mesmo a conta do superusuário, root, ainda não tem nenhuma senha. Para configurar a senha desse usuário digite estas linhas:

```
c:\mysql\bin\mysqladmin -u root password your_password
c:\mysql\bin\mysqladmin -u root -h your_host_name password your_password
```

Você deve descobrir que tarefas anteriores que não exigiam nenhum nome de usuário ou senha agora falharão sem essas informações. Tentar executar

```
c:\mysql\bin\mysqladmin reload
```

ou

```
c:\mysql\bin\mysqladmin shutdown
```

agora falhará.

De agora em diante, você precisará utilizar o flag `-u` e fornecer um nome de usuário e o flag `-p` para informar ao MySQL que você tem uma senha, como neste exemplo:

```
c:\mysql\bin\mysqladmin -u root -p reload
```

Se digitar esse comando, agora o MySQL deve solicitar a senha do usuário root que você acaba de configurar.

Se precisar de informações adicionais, consulte o Web site do MySQL, http://www.mysql.com.

Estamos agora prontos para instalar o Apache sob o Windows. Vamos começar!

Instalando Apache sob Windows

O Apache 1.3 e superior é projetado para ser executado no Windows NT, 2000 e XP. O instalador somente funcionará com a família de processadores x86, como os da Intel. O Apache também é executado no Windows 95 e 98, mas esses não foram testados. Em todos os casos a rede de TCP/IP deve estar instalada. Certifique-se de utilizar a biblioteca Winsock 2 se decidir instalá-la sob Windows 95 ou 98.

Vá para http://httpd.apache.org e faça download do binário Windows da versão atual do Apache 1.3. (O Apache 2.0 utiliza encadeamentos, e algumas bibliotecas PHP externas não são seguras para encadeamentos, então recomendamos que utilize a versão 1.3.)

Fizemos o download deste arquivo: `apache_1.3.31-win32-x86-no_src.msi`. Esse arquivo contém a versão atual (dentro da hierarquia 1.3) para o Windows, sem o código-fonte, empacotado como um arquivo MSI. Os arquivos MSI são o formato de pacote utilizado pelo Windows Installer.

A menos que você tenha um bug realmente ardiloso ou queira contribuir para o esforço de desenvolvimento, é improvável que queira compilar o código-fonte por sua própria conta. Esse único arquivo contém o servidor Apache pronto para ser instalado.

Dê um clique duplo no arquivo que você descarregou para iniciar o processo. O processo de instalação deve parecer familiar para você. Como mostrado na Figura A.5, ele parece semelhante a muitos outros instaladores Windows.

Figura A.5 O instalador Apache é fácil de utilizar.

O programa de instalação apresentará um prompt pedindo o seguinte:

- O nome da rede, o nome do servidor e o endereço de e-mail do administrador. Se estiver construindo um servidor para utilização real, você deverá conhecer as respostas a essas perguntas. Se estiver construindo um servidor para utilização pessoal, as respostas não são particularmente importantes.
- A possibilidade de o Apache ser executado como um serviço. Como com MySQL, normalmente é mais fácil configurá-lo dessa maneira.
- O tipo de instalação. Recomendamos a opção Complete, mas você pode escolher Custom se desejar omitir alguns componentes como a documentação.
- O diretório para instalar o Apache. (O padrão é `C:\Program Files\Apache Group\Apache`.)

Depois de escolher todas essas opções, o servidor Apache será instalado e iniciado.

O Apache ouvirá a porta 80 (a menos que você tenha alterado as diretivas Port, Listen ou BindAddress nos arquivos de configuração) depois que ele iniciar. Para conectar-se ao servidor e acessar a página padrão, carregue um navegador e insira este URL:

```
http://hostlocal/
```

Isso deve responder com uma página de boas-vindas semelhante à mostrada na Figura A.1 e um link para o manual Apache. Se nada acontecer ou você receber um erro, examine o arquivo `error.log` no diretório Dlogs. Se seu host não se conectou com a Internet, talvez você tenha de utilizar este URL:

```
http://127.0.0.1/
```

Esse é o endereço de IP que significa host local.

Se tiver alterado o número da porta a partir de 80, você precisará acrescentar *número_da_porta* no final do URL.

Observe que o Apache *não pode* compartilhar a mesma porta com outra aplicação de TCP/IP.

Você pode iniciar e parar o serviço Apache a partir do seu menu Start: o próprio Apache é adicionado a Programs->Apache HTTP Server. Sob o título "Control Apache Server", você verá que pode iniciar, parar e reiniciar o servidor com os comandos Start, Stop e Restart.

Depois de instalar o Apache, talvez você precise editar os arquivos de configuração que residem no diretório `conf`. Veremos como editar o arquivo de configuração `httpd.conf` quando instalarmos o PHP.

Se precisar ativar o Apache com SSL no Windows, você deve seguir o excelente FAQ em:

http://tud.at/programm/apache-ssl-win32-howto.php3

mas esteja ciente de que isso não é para fracos.

Instalando o PHP para Windows

Para instalar o PHP para Windows, comece fazendo download dos arquivos para PHP5 em:

http://www.php.net.

Você deve fazer o download de dois arquivos para fazer uma instalação sob o Windows. Um deles é o arquivo ZIP contendo PHP (chamado de algo semelhante a `php-5.0.0-Win32.zip`) e o outro é uma coleção de bibliotecas (`pecl-5.0.0-Win32.zip` ou semelhante).

Comece descompactando o arquivo ZIP para o diretório de sua escolha. O local comum é `c:\PHP`, e utilizamos este aqui.

Você pode instalar as bibliotecas PECL descompactando o arquivo PECL para o diretório de extensões. Usando `C:\PHP` como diretório-base, será `C:\PHP\ext\`.

Agora siga estas etapas:

1. No diretório principal você verá um arquivo chamado php.exe e outro chamado php5ts.dll. Esses são os arquivos de que você precisará para executar o PHP como uma CGI. Se quiser executá-lo como um módulo SAPI, você precisará examinar o diretório c:\PHP\sapi e copiar o arquivo DLL relevante para C:\PHP. Se você estiver utilizando o Apache, o arquivo chama-se php5apache.dll, por exemplo.

   Os módulos SAPI são mais rápidos e mais fáceis de manter seguros; a versão CGI permite executar o PHP a partir da linha de comando. Novamente, a escolha é sua.

2. Copie todos os DDls para o diretório de sistema do seu Windows. Isso será C:\winnt\system32 no Windows NT ou 2000, ou C:\windows\system32 no Windows XP.

3. Defina um arquivo de configuração php.ini. O PHP vem com dois arquivos preparados: php.ini-dist e php.ini-recommended. Sugerimos que você utilize o php.ini.-recommended. Faça uma cópia desse arquivo e renomeie-o php.ini. Coloque seu arquivo php.ini no diretório % SYSTEMROOT %. Normalmente isso é c:\winnt ou c:\winnt40 no Windows NT ou 2000, ou c:\windows no Windows XP.

4. Edite seu arquivo php.ini. Há muitas configurações nele, a maioria das quais pode ser ignorada por enquanto. As configurações que você precisa alterar agora são:

   - Altere a diretiva extension_dir para apontar para o diretório em que suas extensões DLLs residem. Na instalação normal, isso será C:\PHP\ext. Portanto, seu php.ini conterá:

     ```
     extension_dir = c:/php/extensions
     ```

   - Configure a diretiva doc_root para apontar para o diretório-raiz que seu servidor Web utiliza. Isso possivelmente será

     ```
     doc_root = "c:/Program Files/Apache Group/Apache/htdocs"
     ```

     se você estiver utilizando o Apache, ou

     ```
     doc_root = "c:/Inetpub/wwwroot"
     ```

     se você estiver utilizando o IIS.

   - Você também pode escolher algumas extensões para executar. Nessa etapa, sugerimos que você simplesmente coloque o PHP em funcionamento; você pode adicionar extensões conforme necessário. A fim de adicionar uma extensão, examine a lista em "Extensões do Windows". Você verá um uma grande quantidade de linhas como:

     ```
     ;extension=php_pdf.dll
     ```

     A fim de ativar essa extensão, você simplesmente pode remover o ponto-e-vírgula no início da linha (e o oposto para desativá-la). Observe que se quiser adicionar outras extensões mais tarde, você deverá reiniciar seu servidor Web depois que alterou o php.ini a fim de que as alterações tenham efeito.

     Feche e salve seu arquivo php.ini. Neste livro, usaremos php_pdf.dll, php_gd2.dll, php_imap.dll e php_mysqli.dll. É preciso retirar os comentários dessas linhas. Você pode descobrir que php_mysqli.dll está faltando. Se estiver, adicione-o da seguinte forma:

     ```
     extension=php_mysqli.dll
     ```

5. Se você estiver utilizando NTFS, certifique-se de que o usuário sob o qual o servidor Web executa tem permissão para ler seu arquivo php.ini.

Adicionando o PHP à configuração do Apache

Você talvez precise editar um dos arquivos de configuração do Apache. Abra o arquivo httpd.conf no seu editor favorito. Em geral, esse arquivo encontra-se no diretório c:\Program Files\Apache Group\Apache\conf\. Procure as seguintes linhas:

```
LoadModule php5_module c:/php/php5apache.dll
AddModule mod_php5.c
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

Se essas linhas não estiverem aí, adicione-as ao final do arquivo, salve-as e reinicie seu servidor Apache.

Adicionando PHP e MySQL ao Microsoft IIS e PWS

Esta seção abrangerá a maneira como adicionar o suporte de PHP e MySQL ao módulo IIS com o ISAPI (php5isapi.dll). (Você também pode instalá-lo como um CGI, mas recomendamos veementemente que você utilize o módulo ISAPI visto que ele é mais rápido.) Ele pressupõe que você seguiu os passos nas seções anteriores. A principal diferença é que a diretiva de configuração doc_root possivelmente é c:/Inetpub/wwwroot.

Depois de fazer isso, você precisa iniciar o Microsoft Management Console (que talvez apareça no seu sistema como o Internet Services Manager). Se você tiver o NT, isso pode ser encontrado no NT4 Option Pack. Se você tiver o 2000 ou o XP, isso pode ser encontrado no Control Panel sob Administrative Tools.

Se você não vir essa opção, pode precisar instalar o IIS a partir dos CDs originais do Windows antes de continuar.

Quando o IIS estiver aberto, você deve ver uma árvore dos serviços no lado esquerdo. Clique com o botão direito do mouse no servidor Web (geralmente chamado Default Web Server) e selecione Properties.

A caixa de diálogo Properties contém bastante informação, mas você precisa mudar apenas algumas coisas. No Home Directory, clique no botão Configuration. Em Application Mappings, selecione Add para adicionar o PHP. Em seguida, forneça um executável. Forneça o caminho completo para a localização do php5isapi.dll (provavelmente c:\php\php5isapi.dll). Na caixa Extension, digite **.php**. Também é preciso marcar a caixa Script Engine se ainda não estiver selecionada. Clique em OK.

Se quiser poder fazer a autenticação HTTP (da qual tratamos neste livro), observe os ISAPI Filters. Selecione Add. É preciso fornecer um nome de filtro (nesse caso, digite **PHP**) e um executável (aqui, forneça o caminho completo para o arquivo php5isapi.dll). Clique em OK. Finalmente, feche a caixa de diálogo Properties clicando em Apply.

Agora, você deve parar (ou verificar se está parado) e reiniciar o servidor Web. Você pode fazer isso do Management Console/Internet Services Manager clicando com o botão direito do mouse no servidor Web e escolhendo Stop. Você pode reiniciar da mesma maneira selecionando Start.

Vamos testar nosso trabalho

Inicie seu servidor Web e teste para assegurar que o PHP está funcionando. Crie um arquivo test.php e adicione a seguinte linha a ele:

```
<? phpinfo( ); ?>
```

Certifique-se de que o arquivo está no diretório-raiz do documento (em geral, :\Program File\Apache Group\Apache\htdocs no Apache ou c:\Inetpub\wwwroot no IIS) e então o posicione no navegador, como a seguir

```
http://localhost/test.php
```

ou

```
http://seu-número-de-ip-aqui/test.php
```

Se vir uma página semelhante àquela mostrada na Figura A.2, você saberá que o PHP está funcionando.

Instalando PEAR

O PHP5 vem com o instalador de pacotes PEAR (PHP Extension and Application Repository). Se estiver utilizando o Windows, vá para a linha de comando e digite:

```
c:\php\go-pear
```

O script `go-pear` fará algumas perguntas simples e diretas sobre o local onde você quer o instalador de pacote e o local onde quer instalar as classes PEAR padrão, em seguida ele fará o download e irá instalá-los para você.

Nessa etapa, você deve ter uma versão instalada do instalador de pacotes PEAR e as bibliotecas básicas PEAR. Depois disso, você pode simplesmente instalar os pacotes digitando:

```
pear install pacote
```

onde *pacote* é substituído pelo nome do pacote que você deseja instalar.

Para obter uma lista dos pacotes disponíveis, digite:

```
pear list-all
```

Para ver o que você instalou atualmente, experimente:

```
pear list
```

Para instalar o pacote de correio MIME utilizado no Capítulo 30, digite:

```
pear install Mail_Mime
```

O pacote DB, mencionado no Capítulo 11, é instalado automaticamente, mas para certificar-se de que tem a versão mais recente você pode digitar:

```
pear list-upgrades
```

Se houver uma versão mais recente disponível, digite:

```
pear upgrade DB
```

Se o procedimento anterior, por qualquer motivo, não funcionar para você, sugerimos que tente fazer download dos pacotes PEAR diretamente. Para isso, acessse http://pear.php.net/packages.php.

Daqui você pode navegar pelos vários pacotes disponíveis. Por exemplo, neste livro utilizamos o Mail_Mime. Clique na página desse pacote e em Download Latest para obter uma cópia. Você precisará descompactar o arquivo zip que descarregou e colocá-lo em algum lugar no seu `include_path`.

Você deve ter um diretório `c:\php\pear` ou um semelhante. Se estiver fazendo o download de pacotes manualmente, sugerimos que coloque os pacotes na árvore de diretórios PEAR. O PEAR tem uma estrutura padrão, assim nossa sugestão é colocar os componentes na localização padrão – esse é o local onde o instalador os teria colocado. Por exemplo, o pacote Mail_Mime pertence à seção Mail; portanto, nesse exemplo, iríamos colocá-lo no diretório `c:\php\pear\Mail`.

Definindo outras configurações

Você pode configurar o PHP e o MySQL com outros servidores Web como o Omni, o HTTPD e o Netscape Enterprise Server. Esses servidores não serão abrangidos neste apêndice, mas você pode localizar informações sobre como configurá-los nos Web sites do MySQL e do PHP:

http://www.mysql.com e http://www.php.net respectivamente.

B

Recursos da Web

ESTE APÊNDICE LISTA ALGUNS DOS MUITOS RECURSOS disponíveis na Web que podem ser utilizados para localizar tutoriais, artigos, notícias e código de PHP de exemplo. Esses são somente alguns dos muitos no mercado. Com certeza, há muito mais do que poderíamos possivelmente listar em um apêndice. E muitos mais que abrem diariamente já que a utilização e a familiaridade com o PHP e MySQL continuam a aumentar entre desenvolvedores Web.

Alguns desses recursos estão em idiomas diferentes como alemão ou francês ou outro diferente do seu idioma nativo. Sugerimos utilizar um tradutor como http://www.systransoft.com para navegar pelo recurso Web no seu idioma nativo.

Recursos de PHP

PHP.Net – *http://www.php.net* – O site original para o PHP. Vá até ele para fazer download das versões binária e de código-fonte do PHP e do manual, a fim de navegar pelos arquivos de lista de mala-direta e manter-se atualizado com as novidades do PHP.

ZEND.com – *http://www.zend.com* – O código-fonte para o mecanismo ZEND que faz o PHP funcionar. Um portal que contém fóruns, artigos, tutoriais e um banco de dados de classes de exemplo e código que você pode utilizar.

PEAR – *http://pear.php.net* – O PHP Extension and Application Repository. Esse é site oficial da extensão do PHP.

PECL – *http://pecl.php.net* – O site irmão do PEAR. PEAR contém classes escritas em PHP; PECL (pronuncia-se "pickle") possui extensões escritas em C. As classes PECL são, às vezes, mais difíceis de instalar, mas têm uma maior funcionalidade e quase sempre são mais poderosas do que as baseadas em PHP.

PHPCommunity – *http://www.phpcommunity.org/* – Um novo site baseado em comunidade.

php|architect – *http://www.phparch.com* – Uma revista sobre o PHP. Fornece artigos gratuitos, ou você pode assinar para receber a revista no formato impresso ou em PDF.

PHP Magazine – *http://www.phpmag.net/* – Outra revista sobre o PHP, também disponível em formato impresso ou eletrônico.

PHPWizard.net – *http://www.phpwizard.net* – A fonte de muitas aplicações interessantes do PHP como phpChat e phpIRC. PHPMyAdmin.Net – *http://www.phpmyadmin.net/* – O lar do famoso front-end da Web baseado em PHP para MySQL.

PHPBuilder.com – *http://www.phpbuilder.com* – O portal para tutoriais de PHP. Nesse site, você encontrará tutoriais sobre quase tudo que puder pensar. O site também tem um fórum e quadro de mensagens para a postagem de perguntas.

DevShed.com – *http://www.devshed.com* – Um site do tipo Portal que oferece excelentes tutoriais sobre PHP, MySQL, Perl e outras linguagens de desenvolvimento.

PX-PHP Code Exchange – *http://px.sklar.com* – Um lugar excelente para iniciar. Aqui você encontrará muitos scripts de exemplo e funções úteis.

PHP Resource – *http://www.php-resource.de* – Uma fonte muito interessante para tutoriais, artigos e scripts. O único "problema" é que o site está em alemão. Recomendamos utilizar um site de serviços de tradução para visualizá-lo.

Weberdev.com – *http://www.WeberDev.com* – Anteriormente conhecido como página de exemplo de PHP da Berber, esse site cresceu e agora é um lugar para tutoriais e código de exemplo. O site destina-se a usuários do PHP e do MySQL e aborda segurança e bancos de dados em geral.

HotScripts.com – *http://www.hotscripts.com* – Uma excelente seleção categorizada de scripts. O site tem scripts em várias linguagens como PHP, ASP e Perl. Ele tem uma coleção excelente de scripts de PHP. Atualizado com muita freqüência. Não deixe de ver se estiver procurando scripts.

PHP Base Library – *http://phplib.sourceforge.net* – Um site utilizado por desenvolvedores para projetos de PHP de larga escala. Oferece uma biblioteca com um uma grande quantidade de ferramentas para uma abordagem alternativa de gerenciamento de sessão, bem como criação de modelos e abstração de banco de dados.

PHP Center – *http://www.php-center.de* – Outro site alemão do tipo portal utilizado para tutoriais, scripts, dicas, truques, anúncios publicitários e mais.

PHP Homepage – *http://www.php-homepage.de* – Outro site alemão sobre o PHP com scripts, artigos, notícias e muito mais. Ele tem uma seção rápida de referência.

PHPIndex.com – *http://www.phpindex.com* – Um inteligente portal de PHP francês com bastante conteúdo relacionado ao PHP. O site contém notícia, FAQs, artigos, listagens de trabalho e muito mais.

WebMonkey.com – *http://www.webmonkey.com* – Um portal com uma grande quantidade de recursos Web, tutoriais do mundo real, código de exemplo e assim por diante. O site abrange projeto, programação, back-end, material de multimídia e muito mais.

PHP Club – *http://www.phpclub.net* – O PHP Club oferece muitos recursos para o iniciante de PHP. O site tem notícia, revisões de livro, código de exemplo, fóruns, FAQs e muitos tutoriais para iniciantes.

PHP Classes Repository – *http://phpclasses.upperdesign.com* – Um site que tem como alvo a distribuição de classes disponíveis gratuitamente escritas no PHP. Não deixe de ver se estiver desenvolvendo código ou se seu projeto for composto de classes. Interessante funcionalidade de pesquisa, então você pode localizar material facilmente.

PHP Resource Index – *http://php.resourceindex.com* – Site de portal para scripts, classes e documentação. O mais interessante sobre esse site é que tudo está bem categorizado, o que pode economizar algum tempo.

PHP Developer – *http://www.phpdeveloper.org* – Ainda outro portal de PHP que fornece notícia, artigos e tutoriais de PHP.

Evil Walrus – *http://www.evilwalrus.com* – Um portal elegante para scripts de PHP.

Oodie.com – *http://www.oodie.com* – Fornece as listas de scripts e provedores de serviço de hospedagem de PHP gratuitos.

Source Forge – *http://sourceforge.net* – Recursos de código-fonte aberto extensíveis. A Source Forge não apenas permite localizar código que pode ser útil, como também fornece acesso a CVS, listas de mala-direta e máquinas para desenvolvedores Open Source (de código-fonte aberto).

Codewalkers – *http://codewalkers.com/* – Contém artigos, críticas de livro, tutoriais e o surpreendente PHP Contest onde você pode ganhar prêmios com suas novas habilidades. Eles patrocinam uma nova competição de código a cada duas semanas.

PHP Developer's Network Unified Forums – *http://forums.devnetwork.net/index.php* – Discussão de tudo relacionado ao PHP.

PHP Kitchen – *http://www.phpkitchen.com/* – Matérias, notícias e defesa do PHP.

Postnuke – *http://www.postnuke.com/* – Um sistema de gerenciamento de conteúdo de PHP freqüentemente utilizado.

PHP Application Tools – *http://www.php-tools.de/* – Um conjunto de classes úteis do PHP.

Recursos específicos de MySQL e SQL

MySQL site – *http://www.mysql.com* – O Web site oficial do MySQL. Fornece excelente documentação, suporte e informações. Não deixe de ver se estiver utilizando o MySQL, especialmente para zona de desenvolvedores e arquivos de lista de mala direta.

SQL Course – *http://sqlcourse.com* – Fornece um tutorial introdutório de SQL com instruções fáceis de entender. Permite praticar o que você aprende sobre um interpretador de SQL on-line. A versão avançada é fornecida em *http://www.sqlcourse2.com*.

SearchDatabase.com – *http://searchdatabase.com* – Portal interessante com uma grande quantidade de informações úteis sobre DBS. Fornece excelentes tutoriais, dicas, white papers, FAQs, revisões e assim por diante. Não deixe de conferir!

Recursos do Apache

Apache Software – *http://www.apache.org* – O lugar para iniciar se você precisar fazer download de códigos-fonte ou binários. O site fornece documentação on-line.

Apache Week – *http://www.apacheweek.com* – Revista semanal on-line que fornece informações essenciais para qualquer pessoa executar um Apache Server ou para qualquer pessoa executar serviços Apache. Obrigatório!

Apache Today – *http://www.apachetoday.com* – Uma fonte diária de notícias e informações sobre o Apache. Os usuários devem fazer assinatura para postarem perguntas.

Desenvolvimento Web

Philip and Alex's Guide to Web Publishing – *http://philip.greenspun.com/panda/* – Um guia irreverente e espirituoso para engenharia de software aplicado à Web. Um dos poucos livros sobre o tópico que foi escrito em co-autoria com um samoiedo.

Índice

Símbolos

Números

A

B

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

U

V

O que há no CD-ROM?

O CD-ROM que acompanha este livro contém versões completas do PHP, do MySQL Apache, várias biblioteca de imagens gráficas, arquivos contendo as listagens de código no livro e, eletronicamente, todo o livro no formato PDF.

Windows

O Apêndice A descreve a configuração do Apache, do MySQL e do PHP em uma plataforma Windows. Incluímos versões Windows desses produtos no CD-ROM.

O Apache 1.3.31 está no diretório `Software\Apache\Windows\Binary`. Dê um clique duplo em `apache_1.3.31-win32-x86-no_src.msi` para carregar o instalador Apache.

Tanto a versão de produção atual do MySQL (4.0 – `mysql-4.0.20-win.zip`) como a versão alfa (5.0 - `mysql-5.0.0a-alpha-win.zip`) estão localizadas no diretório `Software\MySQL\Windows\Binary`. Dê um clique duplo em `SETUP.EXE` para iniciar o programa de instalação do MySQL. Depois siga as instruções no Apêndice A para preparar a instalação MySQL para que você possa acompanhar este livro.

O PHP5 está no diretório `Software\PHP\Binary`. Siga as instruções no Apêndice A para configurar o PHP para esse sistema particular.

Uma coleção de módulos PECL para PHP5 estão disponíveis para uso no diretório `Libraries`.

Linux/Unix

Muitas distribuições do Linux e algumas estações de trabalho do Unix já são configuradas com o MySQL Apache e o PHP. Entretanto, este livro pode não descrever as versões mais recentes. O Apêndice A descreve como configurar o MySQL Apache e o PHP em uma estação de trabalho Linux ou Unix se precisar instalá-los. O código-fonte para MySQL Apache e PHP e instaladores binários para o MySQL no Linux estão incluídos no CD-ROM.

O código-fonte para o Apache 1.3.31 está disponível em Software/Apache/Unix/Source. Se você tiver o GNU `tar` disponível, utilize `httpd-1.3.31.tar.gz`. Caso contrário, utilize `httpd-1.3.31.tar.Z`.

Os instaladores binários do MySQL Max4.0 e 5.0 para Linux estão em `Software\MySQL\Unix\Binary`. Se seu sistema Linux utilizar o gerenciador RPM para instalar software, utilize `MySQL-Max-4.0.20-0.i386.rpm` ou `MySQL-Max-5.0.0-0.I386.RPM` para instalar a parte do servidor MySQL e utilize MySQL-client-4.0.20-0.i386.rpm para instalar a parte do cliente MySQL. Se seu sistema Linux não utilizar o gerenciador RPM para instalar software, utilize `mysql-max-4.0.20-pc-linux-i686.tar.gz` ou `musql-standard-5.0.0-alpha-pc-linux-i686.tar.gz` para instalar as partes cliente e servidor do MySQL.

O código-fonte do MySQL 4.0.20 para Unix está em `mysql-4.0.20.tar.gz`, e para 5.0 em `mysql-5.0.0-alpha.tar.gz`. Os usuários do Solaris devem fazer o download do GNU `tar` para extrair esses arquivos devido a um bug dentro da versão do Solaris do programa `tar`.

O código-fonte para o PHP 5.0 está em `Software/PHP/Unix/Source`.

Uma coleção de módulos PECL para PHP5 estão disponíveis para uso no diretório Libraries.

Acordo de Licença

Abrindo esse pacote, você está concordando em obedecer aos seguintes acordos:

Você não pode copiar nem redistribuir o CD-ROM inteiro como um todo. A cópia e a redistribuição de programas individuais de software no CD-ROM são governadas por termos definidos pelos proprietários dos direitos autorais individuais.

O instalador e o código do(s) autor(es) têm direitos autorais reservados pelo editor e seu(s) autor(es). Programas individuais e outros itens no CD-ROM têm direitos autorais reservados por seus vários autores ou outros proprietários dos direitos autorais. Alguns programas incluídos nesse produto podem ser governados por uma licença Open Source (de código-fonte aberto), que permite redistribuição; veja as informações de licença para cada produto para obter informações adicionais.

Outros programas são incluídos no CD-ROM por permissão especial de seus autores.

Esse software é fornecido "como é" sem qualquer tipo de garantia, explícita ou implicitamente, incluindo mas não limitado às garantias implícitas de comerciabilidade e adequação a um propósito particular. Nem o editor nem seus negociantes ou distribuidores assumem qualquer responsabilidade por qualquer dano, alegado ou real, que advenha do uso desse programa. (Alguns estados não permitem a exclusão de garantias implícitas, então essa exclusão pode não se aplicar a você.)

O guia definitivo para construção de aplicações Web orientadas a bancos de dados com PHP 5 e MySQL

PHP e MySQL
Desenvolvimento Web

Tradução da Terceira Edição

PHP e MySQL são tecnologias populares de código-fonte aberto e são ideais para desenvolvimento rápido de aplicações Web orientadas a bancos de dados. PHP é uma poderosa linguagem de criação de scripts elaborada para permitir aos desenvolvedores criarem aplicações Web repletas de recursos, e o MySQL é um banco de dados rápido e confiável que se integra bem com o PHP e é adequado para aplicações dinâmicas baseadas na Internet.

PHP e MySQL Desenvolvimento Web mostra como utilizar essas ferramentas em conjunto para produzir aplicações Web eficazes e interativas. Ele descreve os fundamentos da linguagem PHP, explica como configurar e trabalhar com um banco de dados MySQL e como utilizar o PHP para interagir com o banco de dados e o servidor.

Este livro prático inclui diversos exemplos que apresentam tarefas comuns como autenticar usuários, construir um carrinho de compras, gerar documentos PDF e imagens dinamicamente, enviar e gerenciar e-mail, facilitar discussões entre usuários, gerenciar conteúdo e conectar-se a serviços Web utilizando XML.

A terceira edição de *PHP e MySQL Desenvolvimento Web* foi totalmente atualizada, revisada e expandida para abranger todo o PHP 5 – incluindo o novo modelo de objetos, o tratamento de exceções aprimorado e o SimpleXML – assim como recursos introduzidos no MySQL 5, como as *procedures* armazenadas.

Luke Welling e **Laura Thomson** são sócios da Tangled Web Design, uma consagrada empresa de desenvolvimento Web, especializada em desenvolver Web sites dinâmicos utilizando o PHP. Welling é um desenvolvedor Web sênior para a MySQL AB, e Thomson é conferencista em programação Web, engenharia de software e comércio eletrônico da Universidade de RMIT em Melbourne, na Austrália. Ambos são freqüentes oradores em conferências importantes sobre código-fonte aberto ao redor do mundo.

"Uma excelente referência para programadores que utilizam PHP e MySQL. Altamente recomendado."
– The Internet Writing Journal

"Um excelente curso introdutório em PHP, e uma ótima cobertura do MySQL".
– WebDynamic

Nível

INICIANTE INTERMEDIÁRIO AVANÇADO

Categoria: Desenvolvimento Web /PHP 5, MySQL 4.1 e 5
Interface: Inglês

O CD-ROM inclui código-fonte para os exemplos do livro, o livro completo em formato PDF e PHP e MySQL para Linux/Unix, Windows e Mac OS X.

Uma empresa Elsevier
www.campus.com.br

ISBN 13 - 978-85-352-1714-8
ISBN 10 - 85-352-1714-2